МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	Кафедра
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

ИНТЕРНЕТ-МАГАЗИН ЭЛЕКТРОНИКИ С ИНТЕЛЛЕКТУАЛЬНЫМ АНАЛИЗОМ ПРЕДПОЧТЕНИЙ КЛИЕНТОВ

Студент _____ 23.06.26 Тельнов Илья Владимирович
(подпись, дата) (ФИО полностью)

Руководитель _____ 23.06.26 Слепцов Н.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)

Заведующий кафедрой _____ Митрохин М.А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой удовлетворительно (протокол заседания ГЭК от 24 июня 2026 г. № 48)

Секретарь ГЭК _____ Гурин Е.И.
(подпись) (фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«07» 04 2026 г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Тельнов Илья Владимирович гр. 22ВВП2 факультета ФВТ

2.Тема работы Интернет-магазин электроники с интеллектуальным анализом предпочтений клиентов

Тема утверждена приказом ПГУ № 347/о от "07" 04 2026 г.

3. Руководитель бакалаврской работы Слепцов Н.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Назначение: серверная часть интеллектуальной образовательной платформы

2) Среда выполнения: Windows 10.

Языковые средства разработки: Python, JavaScript, HTML5, CSS3.

Инструментальные средства: Visual Studio Code, Docker Desktop, Swagger UI

Технологии разработки: объектно-ориентированная парадигма, клиент-серверная архитектура, контейнерезация.

3) Основные функции приложения:

а) регистрация и авторизация пользователя;

б) просмотр каталога товаров с фильтрацией по категориям;

в) поиск товаров по названию и бренду;

г) просмотр и добавление товара в корзину;

е) формирование персонализированных рекомендаций

5. Перечень подлежащих разработке вопросов

Требуется выполнить анализ следующих предметных областей: изучить существующие подходы к разработке интернет-магазинов электроники с системами персонализированных рекомендаций, проанализировать методы сбора и обработки данных пользователей, рассмотреть подходы к построению рекомендательных систем. Необходимо исследовать реализацию REST API на языке Python с использованием фреймворка FastAPI и выбрать подходящие инструменты для валидации данных, работы с базой данных и обработки ошибок

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	52	07.04.26 – 13.04.26	
2.	Анализ требований	13	13.04.26 – 17.04.26	
3.	Выбор технологий, применяемых при проектировании и разработке	52	17.04.26 – 24.04.26	
4.	Проектирование архитектуры	39	24.04.26 – 28.04.26	
5.	Разработка интернет-магазина электроники с интеллектуальным анализом предпочтений клиентов	104	28.04.26 – 17.05.26	
6.	Оформление пояснительной записки	64	17.05.26 – 21.05.26	

Дата выдачи задания "07" апреля 2026 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____ А. Р. Феоктова
24.06.2026 *(подпись, дата)*

Реферат

Бакалаврская работа содержит 109 листа, 15 рисунков, 8 таблиц, 7 источников и 3 приложения.

Цель работы заключается в разработке интернет-магазина электроники, который предоставляет пользователю возможность просматривать каталог товаров с фильтрацией по категориям, цене и брендам, управлять корзиной, а также получать персонализированные рекомендации на основе интеллектуального анализа поведенческих данных и неявных предпочтений.

В работе рассматриваются принципы построения клиент-серверных систем электронной коммерции и типовые паттерны организации взаимодействия между клиентской и серверной частями. В процессе выполнения дипломного проектирования предложена архитектура с выделением REST API как единой точки входа для клиентских приложений. Серверная часть реализована на языке Python с использованием веб-фреймворка FastAPI, обеспечивающего высокую производительность и автоматическую генерацию документации. Модуль интеллектуального анализа предпочтений отвечает за сбор неявных поведенческих, построение user-item матрицы и формирование персонализированных рекомендаций на основе коллаборативной фильтрации и косинусной близости. Разработаны и описаны контракты взаимодействия между клиентом и сервером. Отличительной особенностью решения является сочетание централизованного управления запросами на уровне API с декомпозицией бизнес-логики по функциональным модулям (каталог, корзина, события, рекомендации), что повышает отказоустойчивость и масштабируемость серверной части интернет-магазина электроники.

					ПГУ 1.09.03.01.48.001 ПЗ			
Изм	Лист	№ докум.	Подпи	Дата				
Разраб.		Тельнов И.В.		23.06.26	Интернет-магазин электроники с интеллектуальным анализом предпочтений клиентов Пояснительная записка	Лит.	Лист	Листов
Провер.		Слепцов Н.В.		23.06.26			2	109
Н. Контр.		Бычков А.С.		23.06.26		ФВТ, каф. ВТ гр. 22ВВП2		
Утверд.		Митрохин М.А.		23.06.26				

Оглавление

Введение 5

1 Интернет-магазин электроники с интеллектуальным анализом предпочтений клиентов 7

1.1 Анализ актуальности 7

1.2 Анализ аналогов 8

1.3 Постановка задачи 16

2 Выбор и настройка инструментов разработки 21

2.1 Выбор инструментов разработки 21
2.1.1 Выбор языка и фреймворка для клиентской части 21
2.1.2 Выбор платформы и языка для серверной части 21
2.1.3 Выбор системы хранения данных 21
2.1.4 Выбор библиотек для интеллектуального анализа предпочтений 22

2.2 Выбор среды разработки 22

3 Проектирование системы 24

3.1 Архитектура клиент–серверного приложения 24

3.2 Проектирование структуры базы данных 26

4 Реализация веб–приложения 27

4.1 Описание функционала 27

4.2 Описание работы приложения 31
4.2.1 Начальная загрузка и инициализация 31
4.2.2 Авторизация и регистрация пользователей 31
4.2.3 Просмотр каталога и фильтрация товаров 32
4.2.4 Карточка товара и отслеживание времени просмотра 33

4.3 Реализация серверной части 34
4.3.1 Контроллеры API 34

4.4 Реализация клиентской части 37

4.5. Контейнеризация с Docker 41

5 Тестирование 43

5.1. Регистрация и авторизация 43

5.2. Тестирование каталога и фильтрации 43

5.3. Работа рекомендательной системы 44

5.4. Корзина и сбор поведенческих событий 45

5.5. Тестирование Docker–контейнеризации.....45
6 Безопасность жизнедеятельности.....47
Заключение.....47
Список использованных источников.....50
Приложение А –.....51
Приложение Б –.....56
Приложение В – 104

Введение

В современном мире цифровые технологии играют ключевую роль в развитии электронной коммерции. Интернет-магазины позволяют приобретать товары вне зависимости от географического положения, времени суток и индивидуальных предпочтений покупателей. Особенно актуальным становится переход от традиционных методов продаж к персонализированным решениям, учитывающим поведение и предпочтения каждого клиента.

С развитием технологий искусственного интеллекта и машинного обучения появляются новые возможности для повышения эффективности онлайн-торговли. Одним из перспективных направлений является внедрение интеллектуальных систем анализа предпочтений, способных обрабатывать поведенческие данные пользователя, адаптировать отображение товаров и предоставлять персонализированные рекомендации. Это особенно важно в сегменте электроники, где ассортимент насчитывает сотни моделей со схожими характеристиками, а выбор товара требует от покупателя значительных временных затрат.

Современные интернет-магазины, функционирующие в формате веб-приложений, должны обеспечивать высокую доступность, масштабируемость и устойчивость к нагрузкам. Для этого используются клиент-серверные архитектуры, позволяющие обрабатывать большое количество пользователей одновременно. Такие системы требуют как надежной программной реализации, так и эффективных механизмов сбора и обработки данных.

Одной из ключевых проблем современных интернет-магазинов электроники является информационная перегрузка пользователя. Покупатель вынужден вручную перебирать десятки карточек товаров, тратя время на сравнение малозначимых характеристик. Традиционные методы фильтрации и поиска не учитывают неявные предпочтения пользователя, такие как время просмотра, возвраты к карточке товара, последовательность переходов между

категориями. Кроме того, большинство существующих платформ используют примитивные алгоритмы рекомендаций, а не методы машинного обучения.

Решением данной проблемы является разработка интернет-магазина электроники с интеллектуальным анализом предпочтений клиентов. Такой подход позволяет формировать персонализированные рекомендации на основе неявных поведенческих сигналов, что существенно упрощает процесс выбора товара и повышает конверсию. Преимущества использования подобной системы включают:

- снижение информационной перегрузки – пользователь видит только релевантные товары, сокращая время на поиск подходящей модели;
- персонализированные рекомендации увеличивают вероятность покупки, поскольку пользователю показываются товары, соответствующие его интересам;
- улучшение пользовательского опыта, адаптация интерфейса под предпочтения клиента делает взаимодействие с сайтом более комфортным;
- автоматизация сбора данных, неявные поведенческие сигналы собираются без активного участия пользователя, что обеспечивает высокую полноту данных;
- решение проблемы холодного старта – для новых пользователей система предлагает популярные товары, постепенно адаптируясь к их предпочтениям.

Целью данной работы является разработка интернет-магазина электроники с интеллектуальным анализом предпочтений клиентов на основе неявных поведенческих данных. Клиентская часть системы обеспечивает взаимодействие пользователя с каталогом товаров, корзиной, карточками товаров и получение персонализированных рекомендаций. Серверная часть отвечает за обработку данных, хранение информации о пользователях и товарах, сбор поведенческих событий и работу алгоритмов коллаборативной фильтрации (ALS). Система должна обеспечивать одновременную работу большого числа пользователей и функционировать на различных устройствах с доступом к сети Интернет.

1 Интернет-магазин электроники с интеллектуальным анализом предпочтений клиентов

1.1 Анализ актуальности

Рынок электронной коммерции в сегменте электроники демонстрирует устойчивый рост. По данным аналитических агентств, доля онлайн–продаж электроники и бытовой техники ежегодно увеличивается, при этом ключевым фактором конкурентоспособности интернет–магазинов становится персонализация обслуживания. Пользователи ожидают, что платформа не просто предоставит каталог товаров, но и поможет сделать выбор среди сотен моделей со схожими характеристиками.

Однако большинство существующих интернет–магазинов электроники строятся по классической модели «каталог + корзина», где интеллектуальный анализ предпочтений клиентов либо отсутствует, либо реализован на примитивном уровне (история заказов, популярные товары). Это приводит к ряду проблем:

Информационная перегрузка пользователя. Покупатель вынужден вручную перебирать десятки карточек товаров, тратя время на сравнение малозначимых для него характеристи:

– низкая конверсия. Отсутствие персонализированной выдачи увеличивает вероятность ухода пользователя с сайта без совершения целевого действия;

– потеря потенциальных продаж. Клиент может не увидеть товар, который идеально соответствует его неявно выраженным;

– неэффективность явного сбора данных. Пользователи редко заполняют опросники, ставят оценки или пишут развёрнутые отзывы, что делает традиционные рекомендательные системы малоинформативными.

Особую актуальность приобретают методы неявного сбора поведенческих данных. Эти сигналы формируются естественным образом в процессе взаимодействия пользователя с магазином и не требуют от него дополнительных усилий.

Кроме того, категория «электроника» имеет свою специфику, которая усложняет стандартный подход к анализу предпочтений:

– быстрая смена модельного;

– высокая зависимость спроса от технических характеристик;

– сезонность;

– необходимость учёта совместимости.

Таким образом, актуальность темы обусловлена противоречием между:

– возросшими ожиданиями пользователей к персонализации и скоростью выбора в интернет–магазинах электроники;

– недостаточной развитостью моделей интернет–магазинов, способных в реальном времени анализировать неявные предпочтения и адаптировать выдачу товаров.

Разработка модели интернет–магазина электроники с интеллектуальным анализом предпочтений клиентов позволяет устранить указанное противоречие и повысить эффективность электронной коммерции в данном сегменте.

1.2 Анализ аналогов

В настоящее время существует ряд интернет–магазинов электроники, предоставляющих возможность онлайн–покупок. К наиболее известным относятся такие платформы, как Wildberries, Ozon, М.Видео, Эльдорадо, а также специализированные магазины DNS и Citilink.

Данные платформы предоставляют широкий ассортимент товаров, удобную каталогизацию, фильтрацию по характеристикам и возможность сравнения моделей. Платформа М.Видео, в частности, предлагает систему персональных рекомендаций на основе истории покупок и просмотров, а также интегрированные программы лояльности. Она поддерживает отображение отзывов, рейтингов и сопутствующих товаров.

Однако основным недостатком рассмотренных платформ является ограниченная глубина интеллектуального анализа предпочтений клиентов.

Рекомендательные системы в большинстве случаев работают по шаблонным алгоритмам, не учитывая неявные поведенческие паттерны: время удержания внимания на карточке товара, сценарии сравнения, последовательность переходов между категориями. Обратная связь, как правило, реализуется через явные оценки или историю заказов, что требует активного участия пользователя и не охватывает фазу выбора до покупки.

Существуют также специализированные решения для анализа предпочтений, например, системы класса Recommender Engine. Они используют методы коллаборативной и контентной фильтрации и дают возможность интеграции в интернет–магазины. Однако такие системы, как правило, предлагаются как внешние сервисы, требуют отдельной настройки и не учитывают специфику категории электроника.

Кроме того, многие из существующих решений не обеспечивают достаточную персонализацию в реальном времени и не интегрируют анализ неявных предпочтений непосредственно в процесс подбора ассортимента, ограничиваясь блоками «рекомендуемые товары» на главной странице.

Аналог 1: Раздел электроники Wildberries – Wildberries является крупнейшим маркетплейсом России. Раздел электроники насчитывает более 100 000 товаров от тысяч продавцов.

Сильные стороны:

– огромный ассортимент – представлены практически все категории электроники;

– удобная мобильная версия и приложение;

– система отзывов и рейтингов;

– блок «с этим товаром часто покупают» на основе истории заказов всех пользователей.

Слабые стороны с точки зрения интеллектуального анализа предпочтений:

– рекомендации не персонализированы – блок «часто покупают» одинаков для всех пользователей;

– отсутствует анализ поведения до покупки;

– нет персональных рекомендаций на главной;

– качество рекомендаций сильно зависит от количества отзывов, что создаёт проблему холодного старта для новых товаров.

Оценка по 5 – балльной шкале:

– персонализация: 2/5;

– использование неявных данных: 1/5;

– качество рекомендаций: 3/5;

– скорость адаптации к поведению: 2/5.

Интерфейс сервиса представлен на рисунке 1

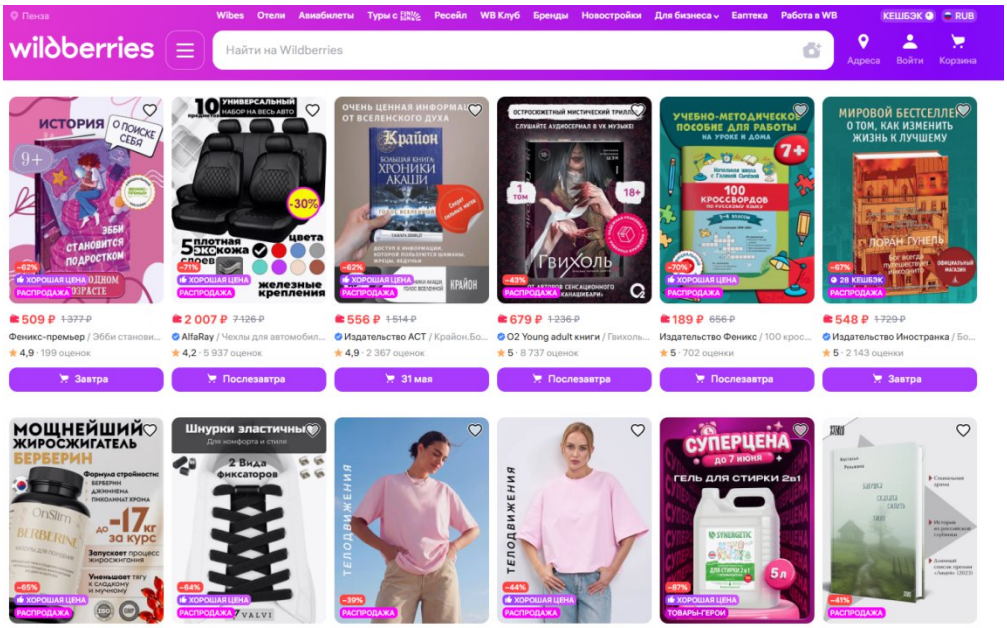


Рисунок 1 – Интерфейс онлайн – магазина Wildberries

Аналог 2: Раздел электроники Ozon – второй по величине маркетплейс России, активно развивающий направление электроники.

Сильные стороны:

– персонализированные рекомендации на главной странице;

– учёт истории просмотров при формировании рекомендаций;

– блок «вы смотрели ранее»;

– система «умные рекомендации» на основе машинного обучения.

Слабые стороны:

– неявные данные используются;

– отсутствует интеграция рекомендаций в карточку товара;

– рекомендации обновляются с задержкой;

– для новых пользователей рекомендации не персонализированы.

Оценка по 5 – балльной шкале:

– персонализация: 3/5;

– использование неявных данных: 2/5;

– качество рекомендаций: 3/5;

– скорость адаптации к поведению: 3/5.

Интерфейс сервиса представлен на рисунке 2.

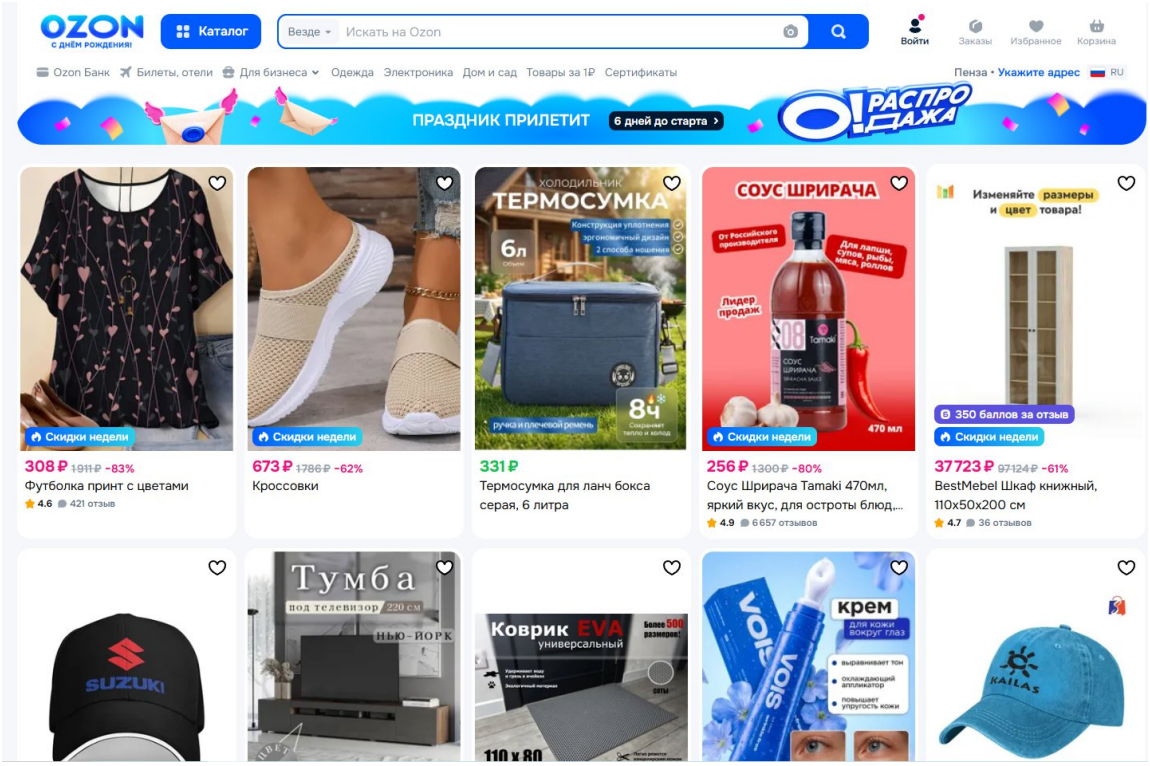


Рисунок 2 – Интерфейс онлайн – магазина Ozon

Аналог 3: М.Видео – крупнейшая российская розничная сеть по продаже электроники и бытовой техники, имеющая развитый онлайн–канал.

Сильные стороны:

– глубокая фильтрация по техническим;

– система сравнения товаров;

– блок «аксессуары к выбранному товару»;

– учёт региональной специфики.

Слабые стороны:

– практически отсутствуют персонализированные рекомендации;

– нет анализа поведения пользователя на сайте (кроме сохранения истории просмотров в текущей сессии).

– отсутствует обучение на данных о просмотрах – после перезагрузки страницы история теряется;

– рекомендации статичны и не адаптируются под пользователя.

Оценка по 5–балльной шкале:

– персонализация: 1/5;

– использование неявных данных: 1/5;

– качество рекомендаций: 2/5;

– скорость адаптации к поведению: 1/5.

Интерфейс сервиса представлен на рисунке 3

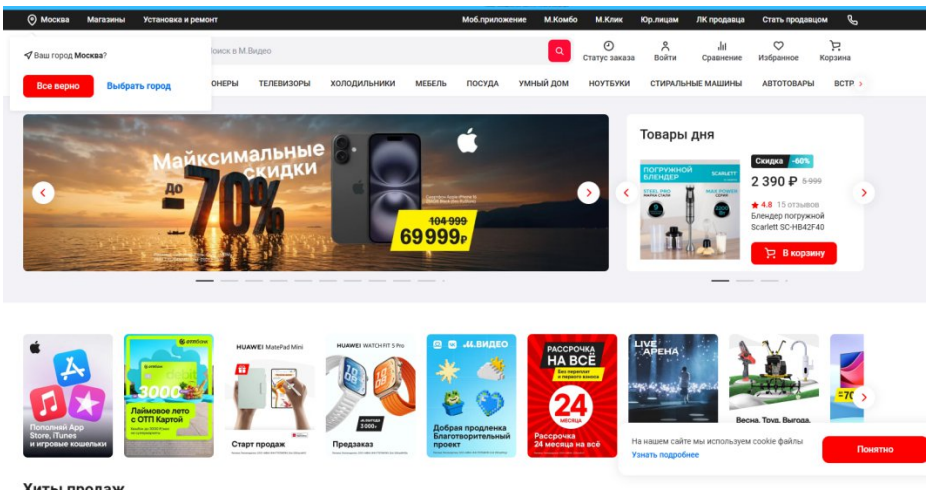


Рисунок 3 – Интерфейс онлайн – магазина М.Видео

Аналог 4: DNS – российская сеть магазинов электроники, ориентированная на технически подкованную аудиторию.

Сильные стороны:

– детальные технические характеристики на карточке товара;

– возможность заказа товаров под заказ;

– блок «похожие товары» на основе характеристик.

Слабые стороны:

– отсутствие персонализации;

– нет сбора поведенческих данных;

– блок «похожие товары» работает по жёстким правилам, а не на основе машинного обучения.

Оценка по 5 – балльной шкале:

– персонализация: 1/5;

– использование неявных данных: 0/5;

– качество рекомендаций: 2/5;

– скорость адаптации к поведению: 0/5.

Интерфейс сервиса представлен на рисунке 4

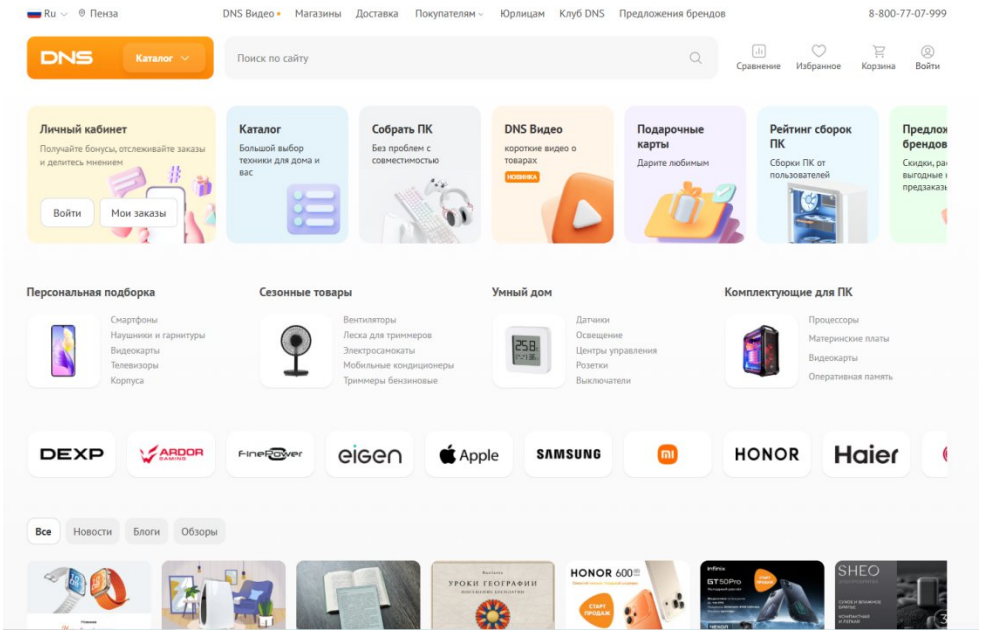


Рисунок 4 – Интерфейс онлайн – магазина DNS

Аналог 5: Citilink – федеральная сеть гипермаркетов электроники и бытовой техники.

Сильные стороны:

– удобный поиск и фильтрация;

– быстрая загрузка страниц;

– программа лояльности с персональными скидками.

Слабые стороны:

– полное отсутствие рекомендательной системы.

– нет блока «похожие товары».

– нет персонализации контента.

Оценка по 5 – балльной шкале:

– персонализация: 0/5;

– использование неявных данных: 0/5;

– качество рекомендаций: 0/5;

– скорость адаптации к поведению: 0/5.

Интерфейс сервиса представлен на рисунке 5

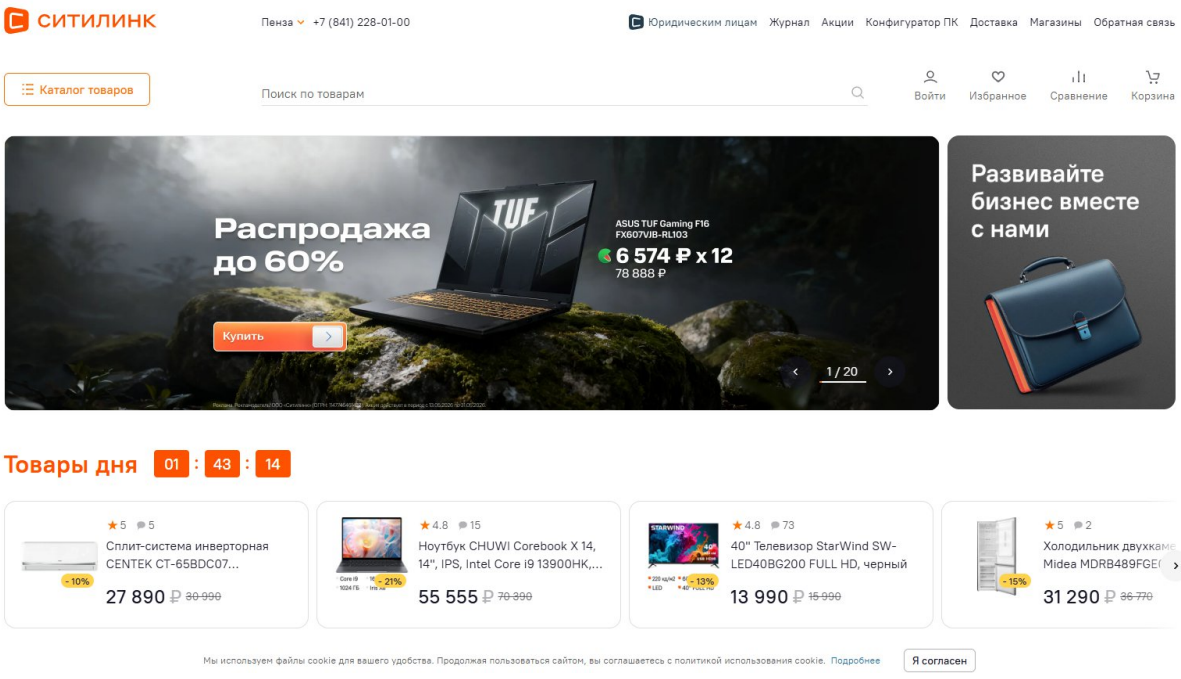


Рисунок 5 – Интерфейс онлайн – магазина

Сравнительный анализ представлен на таблице 1

Таблица 1 – Сравнительный анализ

Аналог	Персонализация	Неявные данные	Качество рекомендаций	Адаптация	Итог
Wildberries	2	1	3	2	2.0
Ozon	3	2	3	3	2.75
М.Видео	1	1	2	1	1.25
DNS	1	0	2	0	0.75
Citilink	0	0	0	0	0

Обобщая анализ аналогов, можно выделить следующие системные недостатки:

– отсутствие полноценного сбора неявных поведенческих данных. Ни один из рассмотренных аналогов не отслеживает такие важные сигналы, как время просмотра карточки товара, возврат к ранее просмотренному товару, последовательность переходов между категориями;

– примитивные алгоритмы рекомендаций. Большинство аналогов используют простейшие. Методы машинного обучения практически не применяются;

– отсутствие персонализации в реальном времени. Рекомендации обновляются с задержкой либо не обновляются вовсе. Поведение пользователя в текущей сессии не влияет на рекомендации;

– нет интеграции рекомендаций в процесс выбора. Рекомендации представлены отдельными блоками на главной странице или в карточке товара, но не влияют на сортировку каталога и не адаптируют интерфейс под пользователя.

Проведённый анализ показывает, что на российском рынке интернет–торговли электроникой отсутствует решение, которое бы:

– объединяло полноценный интернет–магазин с гибким каталогом;

– собирало неявные поведенческие данные;

– применяло методы машинного обучения для формирования рекомендаций;

– обеспечивало персонализацию в реальном времени;

– решало проблему холодного старта для новых пользователей и товаров.

Таким образом, разработка модели интернет–магазина электроники с интеллектуальным анализом предпочтений клиентов является актуальной и востребованной задачей, имеющей как научную новизну, так и практическую ценность.

1.3 Постановка задачи

Требуется разработать модель интернет–магазина электроники, функционирующего в формате веб–приложения, содержащего в себе модуль интеллектуального анализа предпочтений клиентов.

Модель должна отвечать следующим требованиям:

– платформа должна предоставлять пользователю возможность регистрации, авторизации и управления своим профилем;

– пользователь должен иметь доступ к каталогу электроники с возможностью фильтрации по характеристикам;

– каждый товар должен иметь карточку с описанием, характеристиками, ценой и изображениями;

– система должна обеспечивать сбор неявных поведенческих данных: история просмотров, время на странице товара, добавление в корзину и избранное, последовательность переходов между категориями;

– клиентская часть должна обеспечивать отображение персонализированных рекомендаций на основе интеллектуального анализа собранных предпочтений;

– система должна обеспечивать возможность одновременной работы множества пользователей;

– приложение должно функционировать на различных устройствах с доступом к сети Интернет;

– разрабатываемая система должна удовлетворять следующим функциональным требованиям, разбитым по модулям.

Требования к модулю аутентификации и профиля пользователя приведены в таблице 2.

Таблица 2 – Требования к модулю аутентификации и профиля пользователя

ID	Требования
F1	Система должна предоставлять пользователю возможность регистрации с указанием уникального логина и пароля.
F2	Система должна предоставлять возможность авторизации по логину и паролю.
F3	Система должна поддерживать сессию пользователя после авторизации.
F4	Система должна предоставлять возможность выхода из учётной записи.

Требования к программному обеспечению приведены в таблице 3.

Таблица 3 – Требования к программному обеспечению

ID	Требование
F5	Система должна хранить информацию о товарах: название, категория, цена, бренд, описание, URL изображения.

ID	Требование
F6	Система должна отображать список всех товаров на главной странице.
F7	Система должна обеспечивать фильтрацию товаров по категориям (смартфоны, ноутбуки, аксессуары и др.).
F8	Система должна отображать детальную карточку товара при переходе по ссылке.

Требования к модулю корзины и заказов приведены в таблице 4.

Таблица 4 – Требования к модулю корзины и заказов

ID	Требование
F9	Система должна позволять авторизованному пользователю добавлять товары в корзину.
F10	Система должна отображать содержимое корзины с указанием количества и стоимости каждого товара.
F11	Система должна позволять изменять количество товара в корзине.
F12	Система должна позволять удалять товары из корзины.

Требования к модулю сбора поведенческих данных приведены в таблице 5.

Таблица 5 – Требования к модулю сбора поведенческих данных

ID	Требование
F13	Система должна автоматически фиксировать событие просмотра товара при открытии карточки товара.
F14	Система должна фиксировать время, проведённое пользователем на карточке товара.
F15	Система должна фиксировать событие добавления товара в корзину.
F16	Система должна сохранять все события в базе данных с указанием пользователя, товара, типа события и временной метки.

Требования к модулю интеллектуального анализа и рекомендаций приведены в таблице 6.

Таблица 6 – Требования к модулю интеллектуального анализа и рекомендаций

ID	Требование
F17	Система должна на основе накопленных событий обучать модель матричной факторизации (ALS).
F18	Модель должна учитывать вес события: просмотр (вес 1), добавление в корзину (вес 3), покупка (вес 10).
F19	Система должна формировать персональные рекомендации для авторизованного пользователя на основе обученной модели.
F20	Для новых пользователей (отсутствующих в обучающей

ID	Требование
	выборке) система должна рекомендовать популярные товары (решение проблемы холодного старта).
F21	Система должна формировать список похожих товаров на основе item–based коллаборативной фильтрации.
F22	Рекомендации должны возвращаться в виде списка товаров с оценками релевантности (score).

Требования к модулю пользовательского интерфейса приведены в таблице 7.

Таблица 7 – Требования к модулю пользовательского интерфейса

ID	Требование
F23	Интерфейс должен быть адаптивным и корректно отображаться на устройствах с разным разрешением экрана.
F24	На главной странице должен отображаться блок «рекомендуемые для вас» (после авторизации).
F25	В карточке товара должна быть кнопка «похожие товары», отображающая список похожих моделей.
F26	Интерфейс должен информировать пользователя о необходимости авторизации для получения персонализированных рекомендаций

2 Выбор и настройка инструментов разработки

2.1 Выбор инструментов разработки

Разработка современного веб–приложения требует обоснованного выбора технологического стека, от которого зависят такие характеристики системы, как скорость работы, масштабируемость, безопасность и удобство сопровождения кода. В рамках данной работы архитектура проекта разделена на три ключевые составляющие: клиентскую часть , серверную часть и систему управления базами данных. Для каждой из них был проведен анализ требований и выбраны оптимальные инструменты.

2.1.1 Выбор языка и фреймворка для клиентской части

Клиентская часть интернет–магазина электроники должна обеспечивать высокую интерактивность: динамическую подгрузку товаров, фильтрацию, обновление корзины без перезагрузки страницы.

В качестве основного языка клиентской разработки выбран JavaScript. Для построения пользовательского интерфейса используется чистый HTML/CSS/JS без дополнительных фреймворков. Выбор обоснован минимальным порогом входа, высокой производительностью и полным контролем над кодом.

2.1.2 Выбор платформы и языка для серверной части

Серверная часть отвечает за бизнес–логику, обработку запросов от клиента, взаимодействие с БД, сбор поведенческих событий и формирование рекомендаций. Для реализации сервера выбран язык Python и веб–фреймворк FastAPI.

2.1.3 Выбор системы хранения данных

Для хранения информации о пользователях, товарах, корзине и поведенческих событиях используется SQLite – встраиваемая реляционная база данных, хранящаяся в одном файле shop.db. Данный выбор обусловлен простотой настройки и достаточной производительностью для прототипа.

Товары загружаются в базу данных при первом запуске приложения из статического списка, определённого в коде. Это позволяет быстро развернуть демо–версию без необходимости настройки панели администратора.

Структура БД включает таблицы:

– users – пользователи;

– products – товары;

– events – поведенческие события;

– cart – корзина.

2.1.4 Выбор библиотек для интеллектуального анализа предпочтений

Для реализации рекомендательной системы выбрана библиотека implicit с алгоритмом ALS. Выбор обоснован:

– специализация на неявных данных;

– поддержка взвешивания событий;

– высокая скорость работы с разреженными матрицами;

– интеграция с pandas и numpy для обработки данных.

2.2 Выбор среды разработки

Для написания, отладки и сборки проекта используется следующее программное обеспечение:

– основная среда разработки: Visual Studio Code – легковесный и расширяемый редактор кода, оптимизированный для работы с Python, JavaScript, HTML и CSS;

– СУБД и администрирование: DB Browser for SQLite – бесплатная утилита для просмотра и редактирования базы данных shop.db, выполнения SQL–запросов и проверки накопленных поведенческих событий;

– тестирование API: встроенная документация FastAPI – фреймворк FastAPI автоматически генерирует интерактивную Swagger–документацию по адресу http://localhost:8000/docs, позволяющую тестировать все эндпоинты прямо из браузера;

– система контроля версий: Git и GitHub – для отслеживания истории изменений исходного кода;

– контейнеризация: Docker Desktop – для воспроизводимости окружения и упрощения развёртывания проекта.

3 Проектирование системы

3.1 Архитектура клиент–серверного приложения

Разрабатываемая система построена по классической клиент–серверной архитектуре. Клиентская часть (HTML/CSS/JS) представляет собой веб–приложение с динамической подгрузкой данных через AJAX–запросы. Серверная часть работает как REST API. Взаимодействие происходит по протоколу HTTP, данные передаются в формате JSON.

Модуль интеллектуального анализа работает в составе серверной части: при старте обучает ALS–модель на накопленных событиях, затем по запросу возвращает персональные рекомендации.

Клиентская часть представляет собой веб-приложение с динамической подгрузкой данных, реализованное на чистом HTML, CSS и JavaScript без использования дополнительных фреймворков. Данный подход был выбран для обеспечения минимального порога входа, высокой производительности и полного контроля над кодом. Взаимодействие с серверной частью осуществляется через асинхронные AJAX-запросы с использованием стандартного Fetch API, что позволяет обновлять содержимое страницы без её перезагрузки и создавать эффект работы одностраничного приложения (SPA). Пользовательский интерфейс включает в себя следующие функциональные блоки: левая панель с фильтрацией и поиском товаров, центральная область для отображения каталога товаров в виде карточек, а также правая панель для отображения корзины. При клике на товар осуществляется переход на отдельную страницу карточки товара с подробным описанием, характеристиками и блоком похожих товаров.

Серверная часть (backend) разработана на языке Python с использованием веб-фреймворка FastAPI и работает как REST API. FastAPI был выбран благодаря высокой производительности, асинхронной архитектуре, автоматической генерации документации Swagger UI и встроенной валидации данных через Pydantic. В основе серверной части лежит шаблон MVC, где контроллеры

принимают HTTP-запросы от клиентской части, обрабатывают их, взаимодействуют с моделями данных и возвращают клиенту структурированные ответы в формате JSON. Все эндпоинты API разделены по функциональным группам: аутентификация (регистрация и авторизация), работа с товарами (получение списка и детальной карточки), работа с корзиной (добавление, удаление, просмотр), сбор поведенческих событий (просмотры и добавления в корзину) и формирование рекомендаций.

Взаимодействие между клиентом и сервером осуществляется по протоколу HTTP, а информация передается в текстовом формате JSON. Такое решение обеспечивает универсальность и читаемость данных, а также простоту отладки. Клиент отправляет запросы с указанием метода (GET, POST, DELETE), необходимых параметров и, при необходимости, тела запроса в формате JSON. Сервер обрабатывает запрос, выполняет бизнес-логику, взаимодействует с базой данных и возвращает клиенту ответ с соответствующим HTTP-статусом и телом ответа.

Модуль интеллектуального анализа предпочтений работает в составе серверной части и представляет собой отдельный программный модуль, реализованный в файле recommendations.py. При старте приложения модуль автоматически загружает все накопленные поведенческие события из базы данных, строит user-item матрицу, вычисляет косинусную близость между товарами и подготавливает модель к работе. По запросу клиента модуль возвращает персонализированные рекомендации для конкретного пользователя, учитывая его историю просмотров и добавлений в корзину. Для новых пользователей модуль возвращает популярные товары, решая таким образом проблему холодного старта. Кроме того, модуль предоставляет функциональность для получения похожих товаров на основе item-based коллаборативной фильтрации, что используется на странице карточки товара.

Особенностью решения является сочетание централизованного управления запросами на уровне API с декомпозицией бизнес-логики по функциональным модулям. Каждый модуль отвечает за свою область ответственности: модуль

аутентификации — за регистрацию и вход пользователей, модуль товаров — за предоставление информации о товарах, модуль корзины — за управление корзиной, модуль событий — за сбор поведенческих данных, модуль рекомендаций — за формирование персонализированных предложений. Такая архитектура повышает отказоустойчивость и масштабируемость системы, а также упрощает её дальнейшее развитие и добавление нового функционала. по

3.2 Проектирование структуры базы данных

Для хранения информации используется реляционная БД SQLite. Взаимодействие реализовано через ORM SQLAlchemy. Таблица events имеет внешние ключи на users.id и products.id. Вес события определяется типом: view=1, cart=3.

Контекст базы данных объединяет четыре основные сущности: пользователи, товары, поведенческие события и корзина. Каждая сущность представляет собой отдельную таблицу в базе данных, а связи между ними организованы через внешние ключи, что обеспечивает целостность и согласованность данных.

Схема базы данных в графическом виде представляет собой диаграмму «сущность-связь», где таблицы изображены в виде прямоугольников, а связи между ними — в виде линий с указанием типа связи (один-ко-многим). Центральное место в схеме занимает таблица Event, которая связывает пользователей и товары, формируя основу для интеллектуального анализа предпочтений.

4 Реализация веб–приложения

4.1 Описание функционала

Пользователь может:

– просматривать каталог товаров с фильтрацией по категориям;

– искать товары по названию или бренду в реальном времени;

– фильтровать товары по цене и брендам;

– добавлять товары в корзину;

– просматривать детальную карточку товара с описанием, характеристиками и похожими товарами;

– получать персонализированные рекомендации;

– регистрироваться и входить в систему.

Ассортимент товаров загружается из статического списка в коде при первом запуске.

Разработанный интернет–магазин электроники с интеллектуальным анализом предпочтений клиентов представляет собой полноценное веб–приложение. Система предоставляет широкий набор функций, охватывающих все этапы взаимодействия пользователя с интернет–магазином, от регистрации и просмотра каталога до получения персонализированных рекомендаций и управления корзиной.

Система предоставляет пользователям возможность создания учетной записи, просмотра каталога товаров, использование многофакторной фильтрации и поиска, просмотр карточек товара, добавление товаров в корзину,олучения персонализированных рекомендаций.

Варианты использования проектируемой системы представлены на рисунке 6.

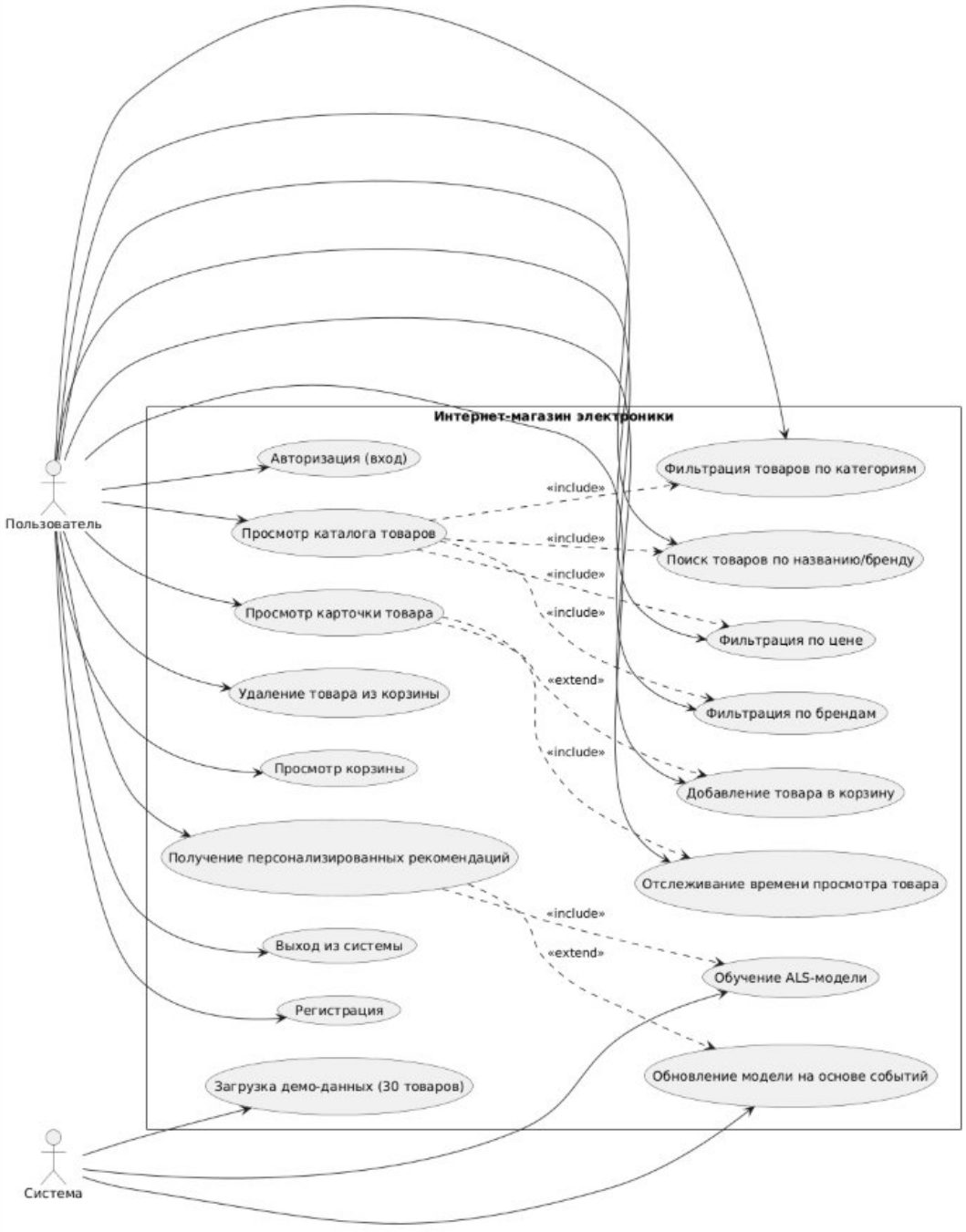


Рисунок 6 – UML–диаграмма вариантов использования проектируемой системы

Варианты деятельности для загрузки демо - данных представлены на рисунке 7

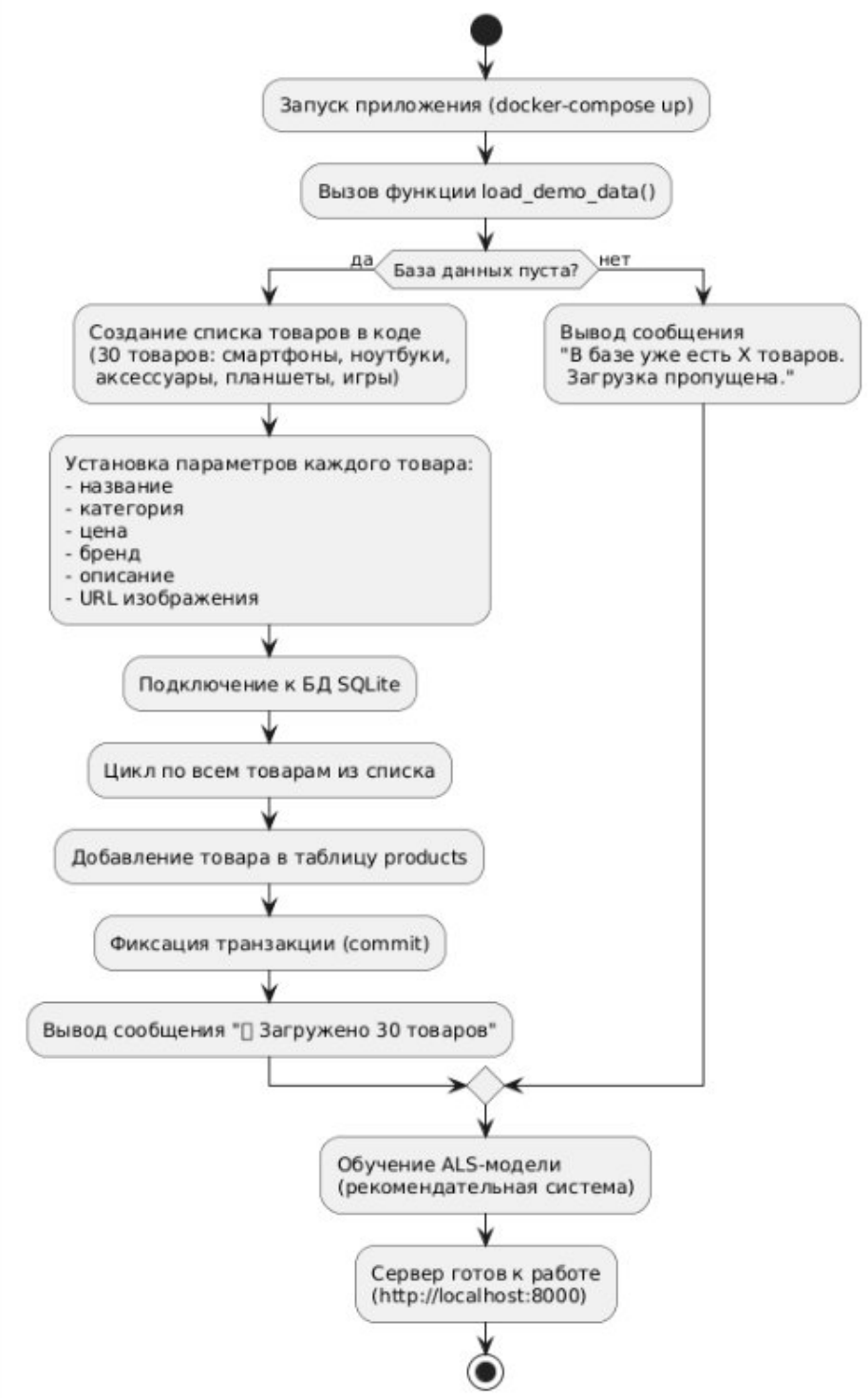


Рисунок 7– UML–диаграмма деятельности для загрузки демо – данных в базу данных

Диаграмма классов представлены на рисунке 8

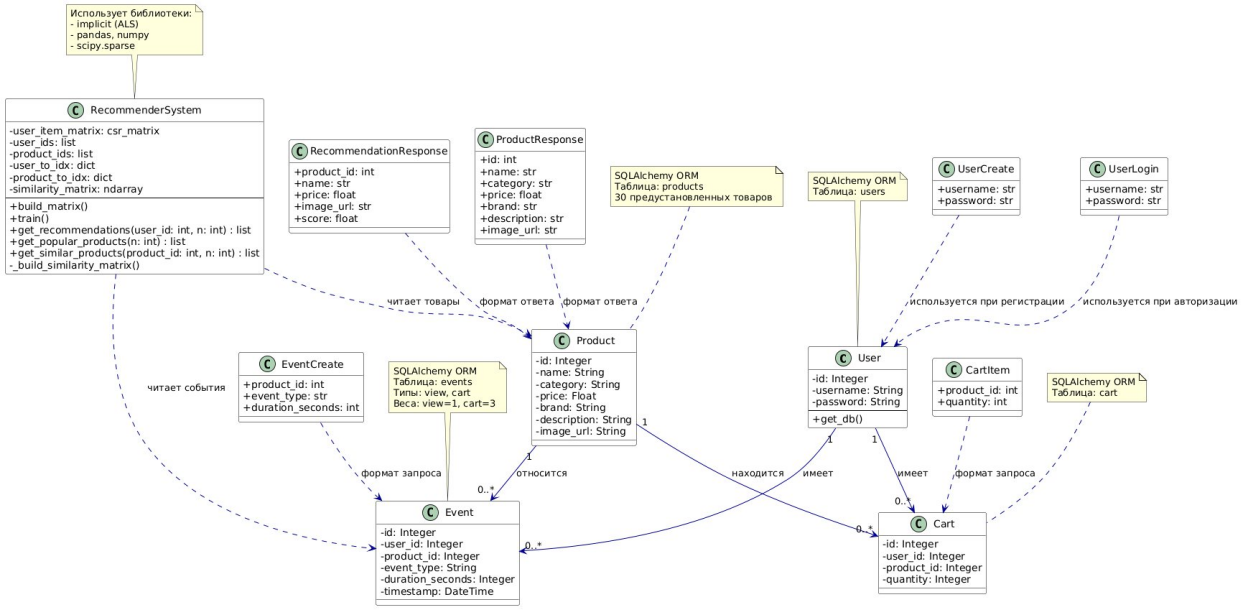


Рисунок 8 – UML–диаграмма классов

Диаграмма классов представлены на рисунке 9

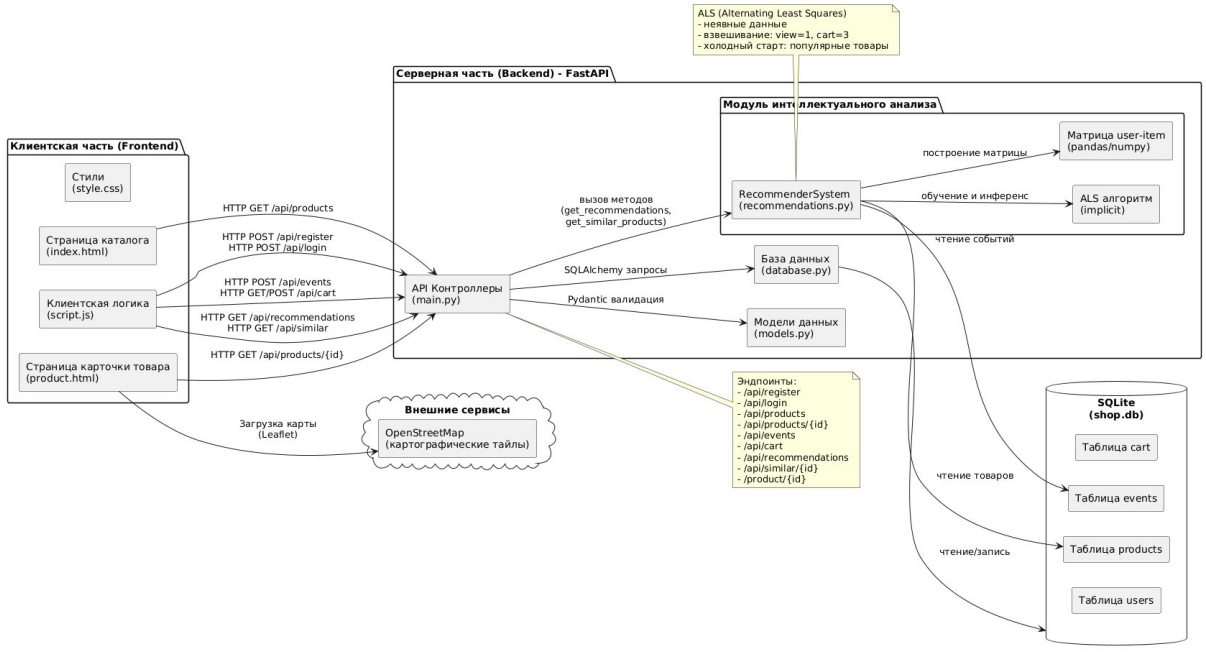


Рисунок 9 – UML–диаграмма компонентов интернет–магазина электроники.

4.2 Описание работы приложения

Пользователь взаимодействует с клиентской частью в браузере. При первом заходе открывается главная страница с каталогом всех товаров. Товары загружаются из SQLite, который при первом запуске автоматически наполняется 30 предустановленными товарами.

Через боковую панель доступна фильтрация по категориям, поиск, выбор брендов и диапазона цен. При изменении любого фильтра список товаров обновляется без перезагрузки страницы.

Для добавления товара в корзину требуется авторизация. При нажатии на кнопку «В корзину» отправляется POST–запрос на /api/cart и POST–запрос на /api/events для сохранения поведенческого события.

Страница карточки товара открывается по клику на любой товар в каталоге по адресу /product/{id}. При открытии карточки запускается таймер, фиксирующий время просмотра – при уходе со страницы это время отправляется на сервер как событие view.

Рекомендательная система обучается при старте сервера. Для авторизованного пользователя кнопка «Для вас» загружает персональные рекомендации.

4.2.1 Начальная загрузка и инициализация

При первом открытии сайта по адресу http://localhost:8000 происходит следующее:

- браузер загружает HTML-страницу index.html, которая содержит структуру магазина;
- загружаются стили style.css, обеспечивающие адаптивный дизайн и визуальное оформление;
- исполняется скрипт script.js, который.

4.2.2 Авторизация и регистрация пользователей

Сценарий регистрации нового пользователя:

Пользователь нажимает кнопку «Регистрация», после чего открывается модальное окно с полями ввода логина и пароля. После заполнения и отправки формы:

- JavaScript формирует POST-запрос к эндпоинту /api/register с JSON-телом {"username": "...", "password": "..."};
- сервер проверяет уникальность логина. Если логин свободен, создаётся новая запись в таблице users;
- при успешной регистрации пользователь получает уведомление и автоматически перенаправляется к форме входа;
- при попытке зарегистрировать существующий логин сервер возвращает ошибку 400, а клиент отображает сообщение «Пользователь уже существует».

Пользователь вводит логин и пароль в форму входа и нажимает «Вход»:

- клиент отправляет POST-запрос на /api/login с учётными данными;
- сервер проверяет наличие пользователя в базе данных и соответствие пароля;
- при успешной аутентификации сервер генерирует JWT-токен, который возвращается клиенту;
- клиент сохраняет токен в localStorage.setItem('token', token).
- интерфейс переключается в авторизованное состояние: кнопки входа/регистрации скрываются, появляется приветствие пользователя и кнопка «Выход»;
- автоматически загружается корзина пользователя через GET /api/cart.

4.2.3 Просмотр каталога и фильтрация товаров

После загрузки страницы пользователь видит каталог всех доступных товаров. Каждый товар представлен в виде карточки.

В левой боковой панели расположены кнопки категорий: «Все товары», «Смартфоны», «Ноутбуки», «Аксессуары», «Планшеты», «Игры и консоли».

Пользователь вводит текст в поле поиска. При каждом изменении текста:

- клиент фильтрует локальную копию списка товаров по вхождению подстроки в название или бренд;
- результаты мгновенно отображаются без отправки запроса на сервер;
- поиск регистронезависимый.

Пользователь вводит минимальное и максимальное значение цены и нажимает «Применить цену»:

- клиент фильтрует товары, оставляя только те, чья цена попадает в указанный диапазон;
- применённые фильтры отображаются в виде бейджей над списком товаров.

В боковой панели динамически формируется список всех брендов, присутствующих в каталоге. Пользователь может выбрать несколько брендов с помощью чекбоксов:

- при выборе/снятии чекбокса обновляется набор выбранных брендов;
- товары отфильтровываются, оставляя только те, бренд которых входит в выбранный набор.

4.2.4 Карточка товара и отслеживание времени просмотра

При клике на карточку товара в каталоге или нажатии на кнопку «Подробнее» пользователь переходит на страницу /product/{id}.

Корзина доступна только авторизованным пользователям и отображается в правой панели на всех страницах.

Добавление товара в корзину:

При нажатии кнопки «В корзину» проверяется наличие JWT-токена. Если пользователь не авторизован, выводится сообщение «Войдите, чтобы добавить товар в корзину».

4.3 Реализация серверной части

4.3.1 Контроллеры API

В файле main.py реализованы эндпоинты приведённые в таблице 8.

Таблица 8 – Требования к модулю сбора поведенческих данных

Метод	Маршрут	Описание
POST	api/register	Регистрация пользователя
POST	api/login	Авторизация, выдача токена
GET	api/products	Список товаров
GET	api/products/{id}	Детальная информация о товаре
POST	api/events	Сохранение поведенческого события
GET	api/cart	Получение корзины
POST	api/cart	Добавление товара в корзину
DELETE	api/cart/{id}	Удаление товара из корзины
GET	api/recommendations	Персональные рекомендации
GET	api/similar/{id}	Похожие товары
GET	product/{id}	HTML–страница карточки товара

Пример эндпоинта регистрации пользователя приведён на листинге 1.

Листинг 1 - Пример эндпоинта регистрации пользователя

```
@app.post("/api/register")
def register(user: UserCreate, db: Session = Depends(get_db)):
    existing = db.query(User).filter(User.username ==
user.username).first()
    if existing:
        raise HTTPException(status_code=400, detail="Пользователь
уже существует")

    new_user = User(username=user.username, password=user.password)
    db.add(new_user)
    db.commit()
    db.refresh(new_user)
```

```
    return {"message": "Регистрация успешна", "user_id":
new_user.id}
```

Пример эндпоинта получения списка товаров с фильтрацией приведён на листинге 2.

Листинг 2 - Пример эндпоинта получения списка товаров с фильтрацией

```
@app.get("/api/products", response_model=List[ProductResponse])
def get_products(category: str = None, db: Session =
Depends(get_db)):
    query = db.query(Product)
    if category:
        query = query.filter(Product.category == category)
    return query.all()
```

Пример эндпоинта добавления товара в корзину приведён на листинге 3.

Листинг 3 - Пример эндпоинта добавления товара в корзину

```
@app.post("/api/cart")
def add_to_cart(item: CartItem, token: str, db: Session =
Depends(get_db)):
    if token not in sessions:
        raise HTTPException(status_code=401, detail="Не
авторизован")

    user_id = sessions[token]

    existing = db.query(Cart).filter(
        Cart.user_id == user_id,
        Cart.product_id == item.product_id
    ).first()

    if existing:
        existing.quantity += item.quantity
    else:
        new_item = Cart(user_id=user_id, product_id=item.product_id,
quantity=item.quantity)
        db.add(new_item)

    db.commit()
    return {"message": "Товар добавлен в корзину"}
```

Для тестирования API используется встроенная документация FastAPI (Swagger UI), доступная по адресу http://localhost:8000/docs.

Модуль recommendations.py реализует класс RecommenderSystem. Основные методы:

– build_matrix() – построение разреженной матрицы user–item из событий с весами;

– train() – обучение ALS–модели через библиотеку implicit;

– get_recommendations() – получение топ–N рекомендаций для пользователя;

– get_popular_products() – популярные товары для холодного старта;

– get_similar_products() – item–based рекомендации.

Пример построения матрицы user-item с весами событий приведён на листинге 4.

Листинг 4 – Пример построения матрицы user-item с весами событий

```
def build_matrix(self):
    """Строит матрицу user-item из неявных событий"""
    db = SessionLocal()
    events = db.query(Event).all()
    db.close()

    if not events:
        return None

    data = []
    for event in events:
        weight = 1 if event.event_type == 'view' else 3
        data.append({
            'user_id': event.user_id,
            'product_id': event.product_id,
            'weight': weight
        })

    df = pd.DataFrame(data)


    self.user_ids = df['user_id'].unique()
    self.product_ids = df['product_id'].unique()
    self.user_to_idx = {uid: i for i, uid in
enumerate(self.user_ids)}
    self.product_to_idx = {pid: i for i, pid in
enumerate(self.product_ids)}


    rows = [self.user_to_idx[uid] for uid in df['user_id']]
```

```
    cols = [self.product_to_idx[pid] for pid in df['product_id']]
    data = df['weight'].values

    self.user_item_matrix = csr_matrix((data, (rows, cols)),
                                       shape=(len(self.user_ids),
len(self.product_ids)))
    return self.user_item_matrix
```

Пример получения персонализированных рекомендаций приведён на листинге 5.

Листинг 5 – Пример получения персонализированных рекомендаций

```
def get_recommendations(self, user_id: int, n: int = 10):
    """Возвращает рекомендации для пользователя"""
    if self.user_item_matrix is None:
        return []

    if user_id not in self.user_to_idx:

        return self.get_popular_products(n)

    user_idx = self.user_to_idx[user_id]
    user_vector =
self.user_item_matrix[user_idx].toarray().flatten()

    interacted = set(np.where(user_vector > 0)[0])

    top_items = np.argsort(user_vector)[-3:][::-1]

    recommended_indices = set()
    for item_idx in top_items:
        if item_idx in interacted:
            continue

        similar = np.argsort(self.similarity_matrix[item_idx])[-
n:][::-1]
        recommended_indices.update(similar[:n])

    return [self.product_ids[idx] for idx in recommended_indices if
idx < len(self.product_ids)]
```

4.4 Реализация клиентской части

Интерфейс реализован на чистом HTML/CSS/JS. Файлы расположены в папке static:

– index.html – главная страница;

– product.html – страница карточки товара;

– style.css – стили;

– script.js – клиентская логика.

Основная структура HTML-страницы.

Пример основной структуры HTML-страницы приведён на листинге 6.

Листинг 6 – Пример основной структуры HTML-страницы

```
<main style="display: flex; gap: 20px; padding: 20px;">
    <!-- Левая панель с фильтрами -->
    <div class="sidebar">
        <h3> Поиск</h3>
        <input type="text" id="search-input" placeholder="Название
или бренд...">

        <h3> Категории</h3>
        <button onclick="setCategory('all')">Все товары</button>
        <button onclick="setCategory('smartphone')">
Смартфоны</button>
        <button onclick="setCategory('laptop')"> Ноутбуки</button>

        <h3>Цена</h3>
        <div class="price-inputs">
            <input type="number" id="price-min" placeholder="от ₽">
            <input type="number" id="price-max" placeholder="до ₽">
        </div>
        <button onclick="applyPriceFilter()">Применить цену</button>
    </div>

    <!-- Центральная область с товарами -->
    <div class="content">
        <div id="products-list" class="products-grid"></div>
    </div>

    <!-- Правая панель с корзиной -->
    <div class="cart-sidebar">
        <h3>Корзина</h3>
        <div id="cart-items"></div>
        <div id="cart-total"></div>
    </div>
</main>
```

Реализована многофакторная фильтрация:

– по категориям;

– по поиску;

– по цене;

– по брендам.

Пример функции фильтрации товаров приведён на листинге 7.

Листинг 7 – Пример функции фильтрации товаров

```
function applyFilters() {
    let filtered = [...allProducts];

    // Фильтр по категории
    if (currentCategory !== 'all') {
        filtered = filtered.filter(p => p.category ===
currentCategory);
    }

    // Фильтр по поиску
    if (searchQuery) {
        filtered = filtered.filter(p =>
            p.name.toLowerCase().includes(searchQuery) ||
            p.brand.toLowerCase().includes(searchQuery)
        );
    }

    // Фильтр по брендам
    if (selectedBrands.size > 0) {
        filtered = filtered.filter(p =>
selectedBrands.has(p.brand));
    }

    // Фильтр по цене
    if (priceMin) filtered = filtered.filter(p => p.price >=
priceMin);
    if (priceMax) filtered = filtered.filter(p => p.price <=
priceMax);

    displayProducts(filtered);
}
```

Корзина обновляется асинхронно. При добавлении товара отправляются два запроса: на /api/events и на /api/cart.

Пример добавления товара в корзину приведён на листинге 8.

Листинг 8 – Пример добавления товара в корзину

```
async function addToCart(productId) {
    if (!token) {
        alert('Войдите, чтобы добавить товар в корзину');
        return;
    }
```

```
    // Отправка события для ML
    await fetch(`/api/events?token=${token}`, {
        method: 'POST',
        headers: {'Content-Type': 'application/json'},
        body: JSON.stringify({
            product_id: productId,
            event_type: 'cart',
            duration_seconds: 0
        })
    });

    // Добавление в корзину
    await fetch(`/api/cart?token=${token}`, {
        method: 'POST',
        headers: {'Content-Type': 'application/json'},
        body: JSON.stringify({product_id: productId, quantity: 1})
    });

    loadCart(); // Обновление отображения корзины
}
```

Страница /product/{id} отображает:

– изображение товара;

– название, бренд, цену, описание, характеристики;

– кнопку добавления в корзину;

– таймер отслеживания времени просмотра;

– блок похожих товаров.

Таймер отслеживания времени просмотра представлен на листинге 9.

Листинг 9 – Пример таймер отслеживания времени просмотра

```
let startTime = Date.now();
let timerInterval;

function startTimer() {
    timerInterval = setInterval(() => {
        if (token) {
            let seconds = Math.floor((Date.now() - startTime) /
1000);
            document.getElementById('seconds').textContent =
seconds;
        }
    }, 1000);
}

function sendViewEvent() {
```

```
    if (token) {
        let duration = Math.floor((Date.now() - startTime) / 1000);
        fetch(`/api/events?token=${token}`, {
            method: 'POST',
            headers: {'Content-Type': 'application/json'},
            body: JSON.stringify({
                product_id: productId,
                event_type: 'view',
                duration_seconds: duration
            })
        });
    }
}

startTimer();
window.addEventListener('beforeunload', sendViewEvent);
```

4.5. Контейнеризация с Docker

Для обеспечения воспроизводимости окружения используется Docker. В корне проекта созданы:

– Dockerfile – описывает сборку образа на основе python:3.11–slim, копирует файлы, устанавливает зависимости, открывает порт 8000;

– docker – compose.yml – описывает сервис shop, монтирует текущую директорию для горячей перезагрузки, пробрасывает порт 8000.

Пример Dockerfile приведён на листинге 10.

Листинг 10 – Пример Dockerfile

```
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY . .
EXPOSE 8000
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Пример Docker-compose.ymlприведён на листинге 11.

Листинг 11 – Пример Docker-compose.yml

```
services:
  shop:
    build: .
    container_name: electronics-shop
```

```
ports:
  - "8000:8000"
volumes:
  - .:/app
environment:
  - PYTHONUNBUFFERED=1
command: uvicorn main:app --host 0.0.0.0 --port 8000 –reload
```

Сборка: docker–compose build. Запуск: docker–compose up. После запуска сервер доступен по адресу http://localhost:8000.

Docker – это платформа для контейнеризации приложений. Она позволяет упаковать приложение со всеми его зависимостями (библиотеками, файлами, настройками) в изолированный контейнер, который можно запустить на любой системе, где установлен Docker. Преимущества инструмента заключаются в:

– воспроизводимость окружения;

– изоляция зависимостей;

– горячая перезагрузка при изменении кода;

– лёгкость демонстрации проекта.

5 Тестирование

5.1. Регистрация и авторизация

При неверно введённом логине или пароле пользователь получает сообщение об ошибке. При корректных данных токен сохраняется в localStorage, интерфейс переключается в авторизованное состояние. Вход в аккаунт представлен на рисунке 10.

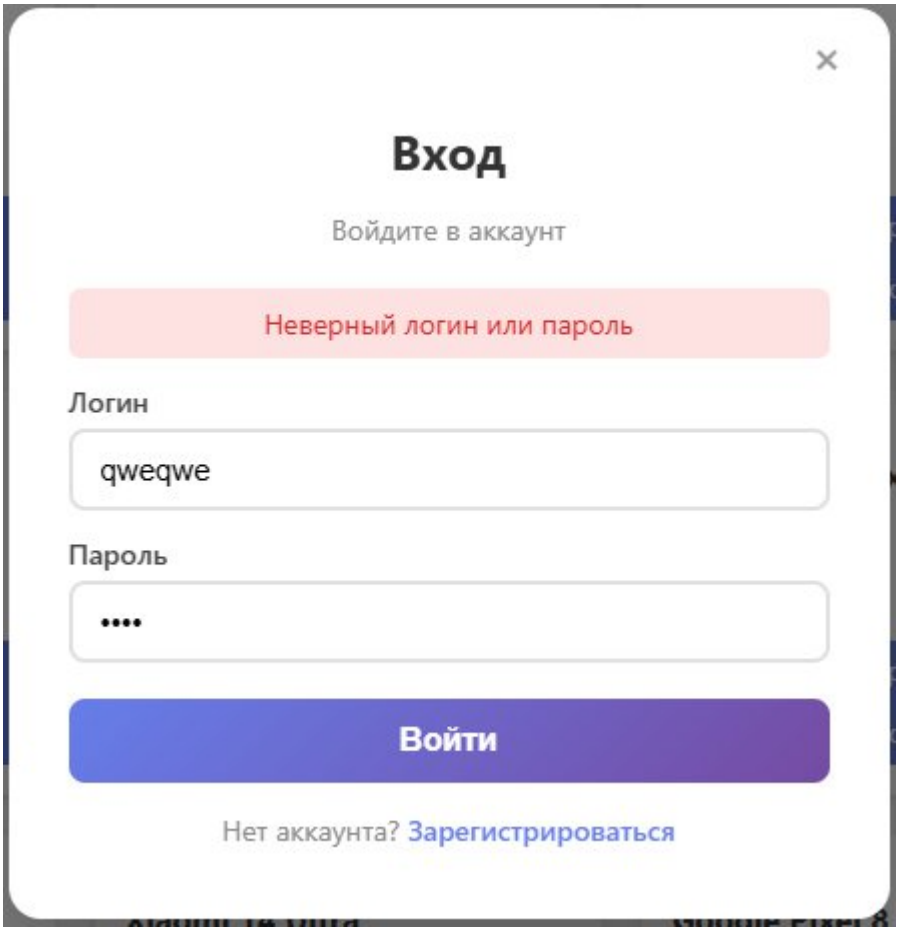


Рисунок 10 – Вход в аккаунт

5.2. Тестирование каталога и фильтрации

Каталог загружается с сервера. При выборе категории «Смартфоны» отображаются только смартфоны. При вводе «iPhone» в поиске отображаются все товары Apple с iPhone в названии. При выборе бренда Apple и ценового диапазона фильтрация работает корректно.

Фильтрация по брендам представлена на рисунке 11.

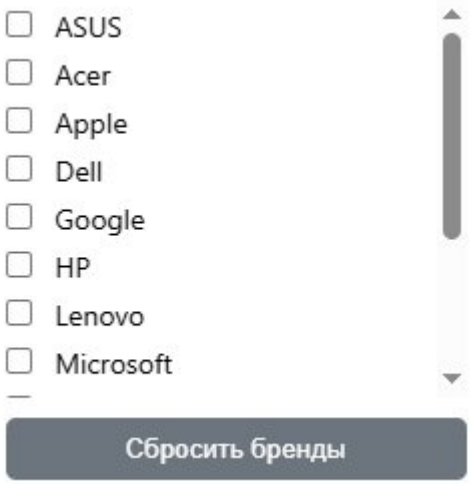


Рисунок 11 – Фильтр по брендам

Фильтрация по типам техники представлена на рисунке 12.

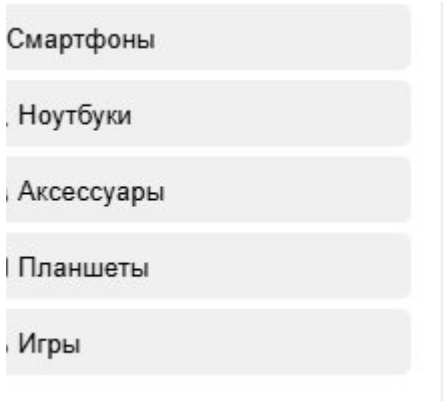


Рисунок 12 – Фильтр по типу техники

5.3. Работа рекомендательной системы

Для нового пользователя кнопка «Для вас» показывает популярные товары (решение холодного старта). После нескольких просмотров и добавлений в

корзину ALS–модель начинает выдавать персонализированные рекомендации. Личные рекомендации представлены на рисунке 13.

Рисунок 13 – Личные рекомендации

5.4. Корзина и сбор поведенческих событий

При добавлении товара в корзину он появляется в правой панели. В таблице events фиксируются события типа cart. При просмотре карточки товара фиксируется событие view с длительностью просмотра. Корзина с товарами представлена на рисунке 14.

Рисунок 14 – Корзина с товарами

5.5. Тестирование Docker–контейнеризации

После выполнения docker – compose up —build контейнер успешно запустился. В логах отобразились сообщения о загрузке 30 товаров и обучении ALS–модели. При открытии http://localhost:8000 интернет–магазин стал доступен. Все функции работали корректно. Горячая перезагрузка при изменении кода представлена на рисунке 15.

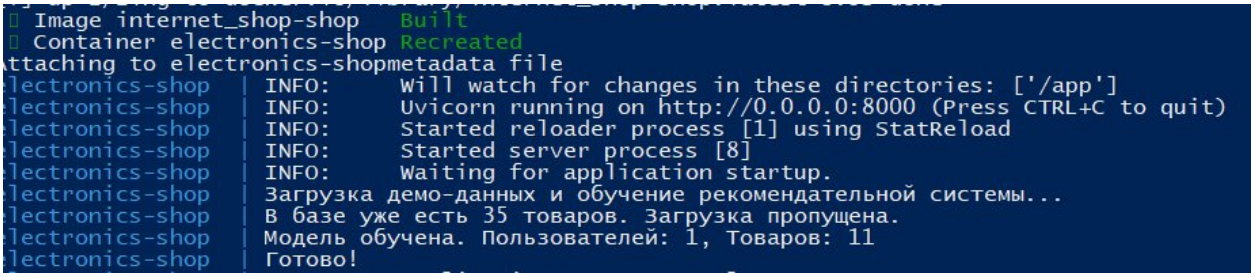


Рисунок 15 – Горячая перезагрузка при изменении кода

6 Безопасность жизнедеятельности

Правильная организация рабочего места пользователя ПЭВМ является важным условием сохранения здоровья и поддержания высокой работоспособности в процессе разработки и эксплуатации программного обеспечения.

Расстояние от монитора до глаз должно составлять 50 –70 см. Экран должен быть расположен перпендикулярно направлению взгляда пользователя, а его центральная точка – находиться на уровне глаз или немного ниже. Это позволяет снизить нагрузку на шейный отдел позвоночника и уменьшить напряжение глазных мышц.

Освещение рабочего места должно быть достаточным и равномерным. При смешанном освещении уровень освещённости должен находиться в пределах 300–500 люкс. Основной поток естественного света следует организовать таким образом, чтобы он падал слева, что исключает появление теней на рабочей поверхности и снижает нагрузку на зрение. Недопустимо наличие прямых бликов на экране монитора, так как они вызывают зрительное утомление и снижают концентрацию внимания.

Расположение компьютера должно исключать направление задней стенки монитора в сторону людей и мест их отдыха. Расстояние от экрана до диванов или кроватей должно составлять не менее 2,5 – 3 метров, что соответствует нормам безопасного расстояния от источников электромагнитного излучения. При необходимости на экран монитора следует устанавливать защитный фильтр, снижающий отражение света и уменьшающий воздействие на глаза.

Важно обеспечить надлежащий тепловой режим работы системного блока. Задняя стенка компьютера не должна быть заблокирована предметами, а сам системный блок не следует размещать вплотную к стене. Это необходимо для

обеспечения нормальной циркуляции воздуха и предотвращения перегрева компонентов, который может привести к сбоям в работе оборудования и сокращению срока его службы.

Обязательным требованием электробезопасности является заземление компьютера. Подключение оборудования должно осуществляться через заземлённую розетку и вилку, что предотвращает накопление статического электричества и снижает риск поражения электрическим током при неисправностях.

Для поддержания безопасных условий труда и предотвращения пожароопасных ситуаций необходимо регулярно проводить влажную уборку помещений. Скопление пыли на внутренних и внешних поверхностях компьютерной техники может привести к перегреву, короткому замыканию и возгоранию. Ежедневная уборка позволяет поддерживать чистоту и минимизировать указанные риски.

Заключение

В ходе выполнения производственной (проектно-технологической) практики была проведена работа по разработке интернет-магазина электроники с интеллектуальным анализом предпочтений клиентов.

В рамках практики были решены следующие задачи: проведён обзор существующих интернет-магазинов электроники Wildberries, Ozon, М.Видео, DNS. Выявлены недостатки: отсутствие полноценного сбора неявных поведенческих данных, примитивные алгоритмы рекомендаций, отсутствие персонализации в реальном времени, нерешённая проблема холодного старта. Обоснована актуальность разработки собственного решения.

Формулирование требований. Определены функциональные требования к системе: регистрация и авторизация пользователей, каталог с фильтрацией, корзина, карточка товара, сбор поведенческих событий, персонализированные рекомендации на основе ALS-модели, контейнеризация с помощью Docker.

Выбор технологического стека. Обоснован выбор инструментов: язык Python и веб-фреймворк FastAPI для серверной части; HTML/CSS/JS для клиентской части; SQLite для хранения данных; библиотека implicit с алгоритмом ALS для рекомендательной системы; Docker для контейнеризации; Visual Studio Code в качестве среды разработки.

Проектирование системы. Разработана архитектура клиент-серверного приложения, спроектирована структура базы данных. Созданы UML-диаграммы: вариантов использования, последовательности, развёртывания, деятельности, компонентов и классов.

Реализация системы. Разработаны: Серверная часть на FastAPI с эндпоинтами для регистрации, авторизации, работы с товарами, корзиной,

событиями и рекомендациями, модуль интеллектуального анализа предпочтений на основе ALS-модели с решением проблемы холодного старта

Список использованных источников

1. Документация FastAPI. — URL: https://fastapi.tiangolo.com/ (дата обращения: 10.05.2026).
2. Документация SQLAlchemy. — URL: https://www.sqlalchemy.org/ (дата обращения: 10.05.2026).
3. Документация Docker. — URL: https://docs.docker.com/ (дата обращения: 10.05.2026).
4. Hagan, R. AI Recommendation System API [Электронный ресурс]. — GitHub, 2025. — URL: https://github.com/TheHiddenDeveloper/AI-Recc-System (дата обращения: 12.05.2026).
5. Алгоритм ALS для коллаборативной фильтрации [Электронный ресурс]. — Apache Spark Documentation. — URL: https://apache.googlesource.com/spark/ (дата обращения: 12.05.2026).
6. Personalized Product Recommendations using ALS-based Collaborative Filtering [Электронный ресурс]. — KCI (Korean Citation Index), 2025. — URL: https://www.kci.go.kr/ (дата обращения: 12.05.2026).
7. Bala Priya C. A Gentle Introduction to Docker for Python Developers [Электронный ресурс]. — KDnuggets, 2025. — URL: https://www.kdnuggets.com/ (дата обращения: 12.05.2026).

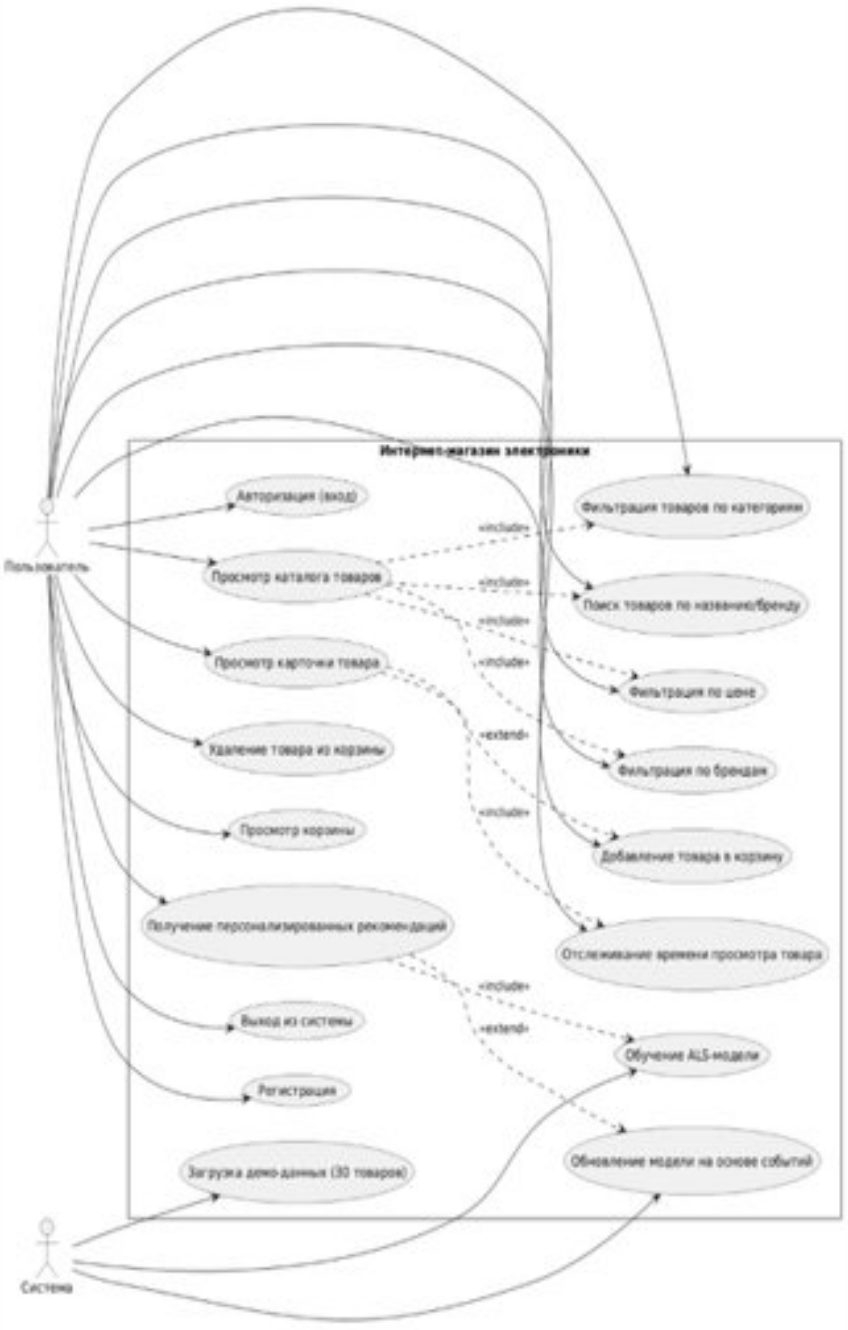
Приложение А

(обязательное)

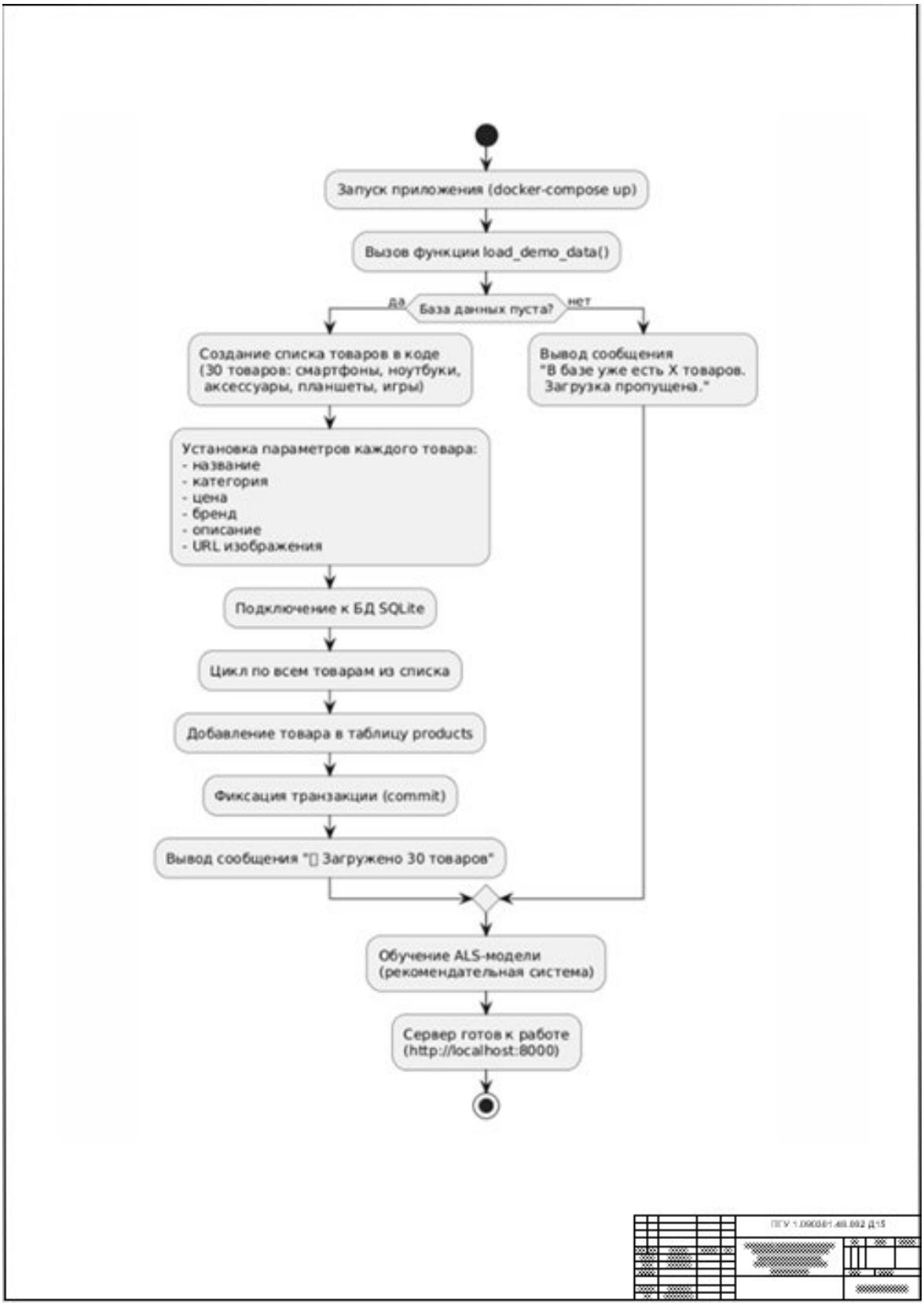
Интернет-магазин электронной техники с интеллектуальным анализом предпочтений клиентов

UML – диаграммы

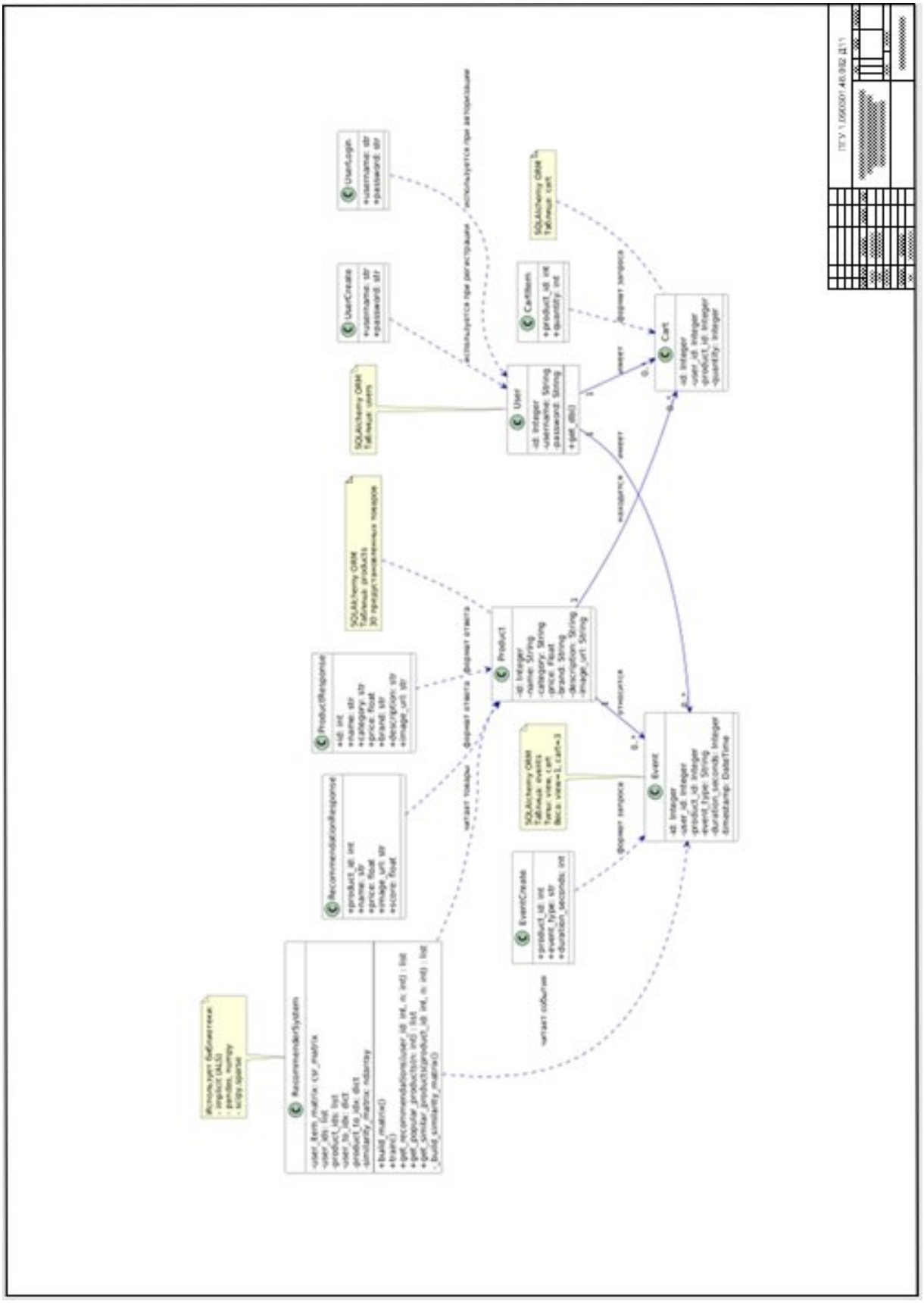
UML – диаграмма вариантов использования


Интернет-магазин электроники
Пользователь
Авторизация (вход)
Просмотр каталога товаров
Просмотр карточки товара
Удаление товара из корзины
Просмотр корзины
Получение персонализированных рекомендаций
Выход из системы
Регистрация
Загрузка демо-данных (30 товаров)
Фильтрация товаров по категории
Поиск товаров по названию/бренду
Фильтрация по цене
Фильтрация по брендам
Добавление товара в корзину
Отслеживание времени просмотра товара
Обучение ALS-модели
Обновление модели на основе событий
Система
«include»
«include»
«include»
«include»
«extend»
«include»
«include»
«extend»

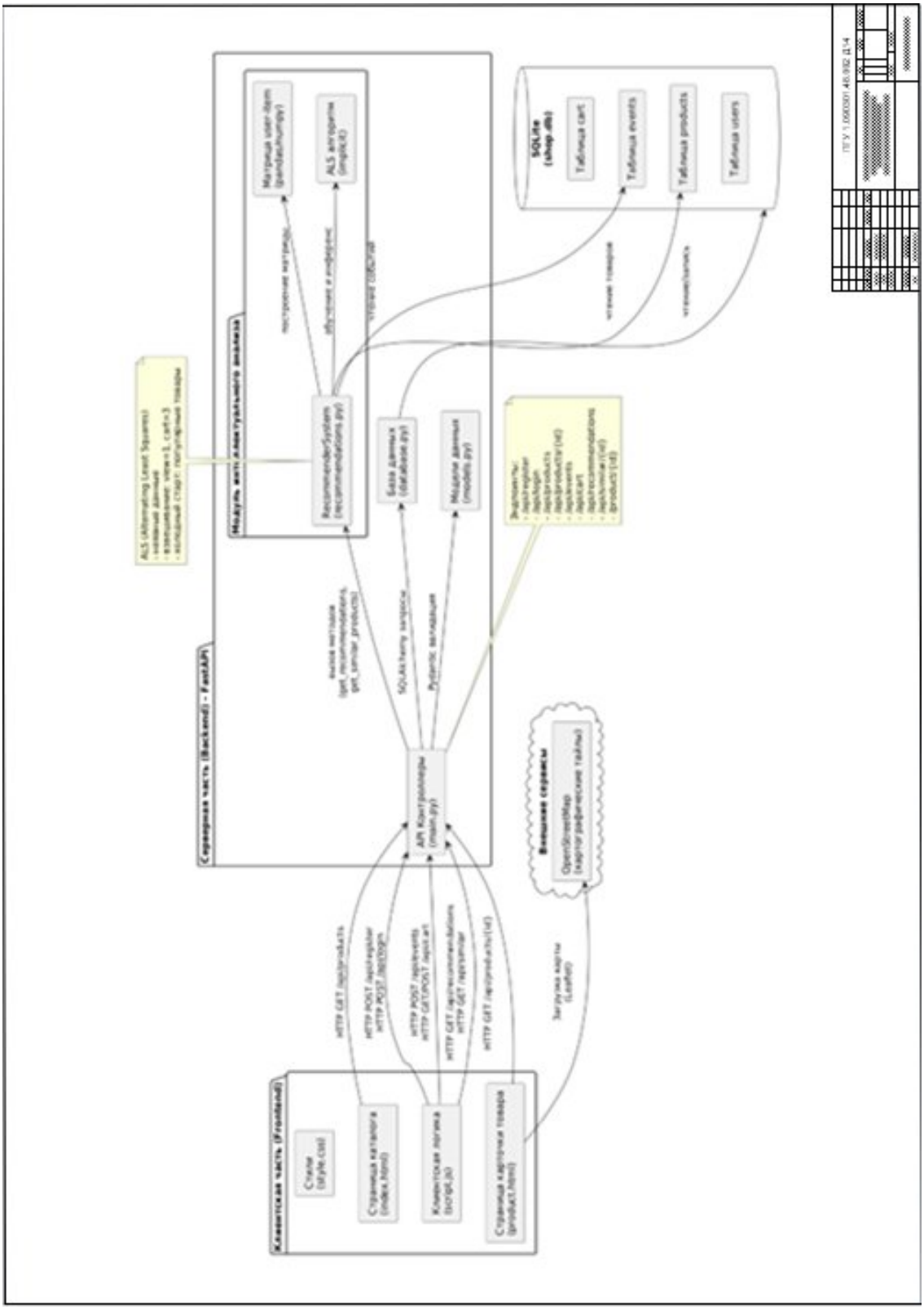
UML – диаграмма деятельности для загрузки демо – данных в базу данных



UML – диаграмма классов



UML – диаграмма компонентов интернет-магазина электроники



Приложение Б

(обязательное)

Интернет-магазин электронной техники с интеллектуальным анализом предпочтений клиентов

Листинг программы

Файл «main.py»

```
from fastapi import FastAPI, HTTPException, Depends, Request
from fastapi.responses import HTMLResponse
from fastapi.staticfiles import StaticFiles
from fastapi.templating import Jinja2Templates
from sqlalchemy.orm import Session
from typing import List
import random
from datetime import datetime, timedelta

from database import get_db, User, Product, Event, Cart
from models import UserCreate, UserLogin, ProductResponse,
EventCreate, CartItem, RecommendationResponse
from recommendations import recommender

app = FastAPI(title="Интернет-магазин электроники")

app.mount("/static", StaticFiles(directory="static"), name="static")
templates = Jinja2Templates(directory="static")

sessions = {}

@app.on_event("startup")
def startup():
    print("Загрузка демо-данных и обучение рекомендательной
системы...")
    load_demo_data()
    generate_test_events()
    recommender.train()
    print("Готово!")

def generate_test_events():
    """Генерирует тестовые события, чтобы рекомендации работали
сразу"""
    db = next(get_db())

    users = db.query(User).all()
    if not users:
        print("Нет пользователей для генерации тестовых событий")
        return

    existing = db.query(Event).first()
    if existing:
        print("События уже есть, пропускаем генерацию")
        return

    products = db.query(Product).all()
    if not products:
        return

    print("Генерация тестовых событий...")
```

```
    events = []
    for user in users:
        num_views = random.randint(5, 10)
        viewed_products = random.sample(products, min(num_views,
len(products)))

        for product in viewed_products:
            duration = random.randint(10, 60)
            events.append(Event(
                user_id=user.id,
                product_id=product.id,
                event_type='view',
                duration_seconds=duration,
                timestamp=datetime.now() -
timedelta(days=random.randint(0, 3))
            ))

        if random.random() > 0.5:
            cart_items = random.sample(viewed_products,
min(random.randint(1, 3), len(viewed_products)))
            for product in cart_items:
                events.append(Event(
                    user_id=user.id,
                    product_id=product.id,
                    event_type='cart',
                    duration_seconds=0,
                    timestamp=datetime.now() -
timedelta(days=random.randint(0, 1))
                ))

    if events:
        db.add_all(events)
        db.commit()
        print(f"Сгенерировано {len(events)} тестовых событий")
    else:
        print("Не удалось сгенерировать события")

def load_demo_data():
    db = next(get_db())

    if db.query(Product).count() > 0:
        print(f"В базе уже есть {db.query(Product).count()} товаров.
Загрузка пропущена.")
        return

    products = [
        Product(name="iPhone 15 Pro", category="smartphone",
price=99900, brand="Apple",
                description="Флагманский смартфон с чипом A17 Pro,
128GB.",
                image_url="/static/images/15pro.jpg"),
```

```
        Product(name="iPhone 15 Pro Max", category="smartphone",
price=119900, brand="Apple",
                description="6.7-дюймовый дисплей, 256GB, лучшая
камера.",
                image_url="/static/images/15promax.jpg"),
        Product(name="iPhone 15", category="smartphone",
price=79900, brand="Apple",
                description="A16 Bionic, 48MP камера, Dynamic
Island, 128GB.",
                image_url="/static/images/15.jpg"),
        Product(name="Samsung Galaxy S24 Ultra",
category="smartphone", price=109900, brand="Samsung",
                description="S Pen в комплекте, 200MP камера,
256GB.",
                image_url="/static/images/s24.jpg"),
        Product(name="Samsung Galaxy S24 Plus",
category="smartphone", price=89900, brand="Samsung",
                description="6.7 дюймов Dynamic AMOLED 2X, 12GB RAM,
256GB.",
                image_url="/static/images/s24plus.jpg"),
        Product(name="Samsung Galaxy A54", category="smartphone",
price=34900, brand="Samsung",
                description="Бюджетный флагман, 6.4 дюйма Super
AMOLED, 128GB.",
                image_url="/static/images/a54.jpg"),
        Product(name="Xiaomi 14", category="smartphone",
price=59900, brand="Xiaomi",
                description="Leica камера, Snapdragon 8 Gen 3,
256GB.",
                image_url="/static/images/14.jpg"),
        Product(name="Xiaomi 14 Ultra", category="smartphone",
price=89900, brand="Xiaomi",
                description="1-дюймовый сенсор Sony, 512GB, кожаный
корпус.",
                image_url="/static/images/14ultra.jpg"),
        Product(name="Google Pixel 8 Pro", category="smartphone",
price=79900, brand="Google",
                description="Лучшая камера среди Android, 7 лет
обновлений.",
                image_url="/static/images/8pro.jpg"),
        Product(name="Nothing Phone 2", category="smartphone",
price=54900, brand="Nothing",
                description="Уникальный дизайн с Glyph-подсветкой,
12GB RAM.",
                image_url="/static/images/no2.jpg"),

        Product(name="MacBook Pro 14", category="laptop",
price=199900, brand="Apple",
                description="Чип M3 Pro, 18GB RAM, 512GB SSD.",
                image_url="/static/images/pro14.jpg"),
        Product(name="MacBook Pro 16", category="laptop",
price=249900, brand="Apple",
```

```
                description="Чип M3 Max, 36GB RAM, 1TB SSD.",
                image_url="/static/images/pro16.jpg"),
        Product(name="MacBook Air 15", category="laptop",
price=139900, brand="Apple",
                description="Чип M2, 8GB RAM, 256GB SSD.",
                image_url="/static/images/air15.jpg"),
        Product(name="Dell XPS 15", category="laptop", price=159900,
brand="Dell",
                description="OLED 3.5K, Intel Core i9, 32GB RAM, 1TB
SSD.",
                image_url="/static/images/15.jpg"),
        Product(name="ASUS ROG Zephyrus G14", category="laptop",
price=129900, brand="ASUS",
                description="Ryzen 9, RTX 4060, 16GB RAM, 1TB SSD.",
                image_url="/static/images/15.jpg"),
        Product(name="Lenovo Legion 5 Pro", category="laptop",
price=119900, brand="Lenovo",
                description="Ryzen 7, RTX 4060, 16GB RAM, 512GB
SSD.",
                image_url="/static/images/15.jpg"),
        Product(name="HP Omen 16", category="laptop", price=109900,
brand="HP",
                description="Intel Core i7, RTX 4050, 16GB RAM,
512GB SSD.",
                image_url="/static/images/15.jpg"),
        Product(name="Acer Predator Helios 16", category="laptop",
price=149900, brand="Acer",
                description="Intel Core i9, RTX 4070, 32GB RAM, 1TB
SSD.",
                image_url="/static/images/15.jpg"),

        Product(name="iPad Pro 11", category="tablet", price=99900,
brand="Apple",
                description="Чип M2, 128GB, Liquid Retina 120Hz.",
                image_url="/static/images/15.jpg"),
        Product(name="iPad Pro 12.9", category="tablet",
price=129900, brand="Apple",
                description="Чип M2, 256GB, Liquid Retina XDR.",
                image_url="/static/images/15.jpg"),
        Product(name="iPad Air", category="tablet", price=69900,
brand="Apple",
                description="Чип M1, 64GB, 10.9 дюймов.",
                image_url="/static/images/15.jpg"),
        Product(name="Xiaomi Pad 6", category="tablet", price=34900,
brand="Xiaomi",
                description="Snapdragon 870, 11 дюймов 144Hz,
128GB.",
                image_url="/static/images/15.jpg"),
        Product(name="Samsung Galaxy Tab S9", category="tablet",
price=79900, brand="Samsung",
                description="Dynamic AMOLED 2X, S Pen в комплекте,
128GB.",
```

```
                image_url="/static/images/15.jpg"),

        Product(name="AirPods Pro 2", category="accessory",
price=24900, brand="Apple",
                description="Активное шумоподавление, адаптивная
прозрачность.",
                image_url="/static/images/15.jpg"),
        Product(name="AirPods Max", category="accessory",
price=54900, brand="Apple",
                description="Полноразмерные наушники премиум-
класса.",
                image_url="/static/images/15.jpg"),
        Product(name="Samsung Buds2 Pro", category="accessory",
price=15900, brand="Samsung",
                description="24-bit Hi-Fi звук, интеллектуальное
шумоподавление.",
                image_url="/static/images/15.jpg"),
        Product(name="Apple Watch Ultra 2", category="accessory",
price=79900, brand="Apple",
                description="49mm титановый корпус, 36 часов
работы.",
                image_url="/static/images/15.jpg"),
        Product(name="Apple Watch Series 9", category="accessory",
price=45900, brand="Apple",
                description="41mm Always-On Retina, S9 чип, ЭКГ.",
                image_url="/static/images/15.jpg"),
        Product(name="Samsung Galaxy Watch 6", category="accessory",
price=29900, brand="Samsung",
                description="44mm Sapphire Crystal, BioActive
Sensor.",
                image_url="/static/images/15.jpg"),
        Product(name="Xiaomi Smart Band 8", category="accessory",
price=3990, brand="Xiaomi",
                description="1.62 дюйма AMOLED, 16 дней работы.",
                image_url="/static/images/15.jpg"),

        Product(name="PlayStation 5 Slim", category="gaming",
price=54900, brand="Sony",
                description="Цифровая версия, 825GB SSD, до 4K
120fps.",
                image_url="/static/images/15.jpg"),
        Product(name="PlayStation 5 с дисководом",
category="gaming", price=64900, brand="Sony",
                description="Стандартная версия, 825GB SSD, 4K Blu-
ray.",
                image_url="/static/images/15.jpg"),
        Product(name="Xbox Series X", category="gaming",
price=55900, brand="Microsoft",
                description="12 терафлопс, 1TB SSD, 4K при 120fps.",
                image_url="/static/images/15.jpg"),
        Product(name="Xbox Series S", category="gaming",
price=29900, brand="Microsoft",
```

```
                description="Цифровая версия, 512GB SSD, до 1440p
120fps.",
                image_url="/static/images/15.jpg"),
        Product(name="Nintendo Switch OLED", category="gaming",
price=34900, brand="Nintendo",
                description="Гибридная консоль, 7-дюймовый OLED-
экран, 64GB.",
                image_url="/static/images/15.jpg"),
    ]

    for p in products:
        db.add(p)

    db.commit()
    print(f"Загружено {len(products)} товаров в базу данных")

@app.post("/api/register")
def register(user: UserCreate, db: Session = Depends(get_db)):
    existing = db.query(User).filter(User.username ==
user.username).first()
    if existing:
        raise HTTPException(status_code=400, detail="Пользователь
уже существует")

    new_user = User(username=user.username, password=user.password)
    db.add(new_user)
    db.commit()
    db.refresh(new_user)

    return {"message": "Регистрация успешна", "user_id":
new_user.id}

@app.post("/api/login")
def login(user: UserLogin, db: Session = Depends(get_db)):
    db_user = db.query(User).filter(User.username == user.username,
User.password == user.password).first()
    if not db_user:
        raise HTTPException(status_code=401, detail="Неверный логин
или пароль")

    token = f"user_{db_user.id}"
    sessions[token] = db_user.id

    return {"token": token, "user_id": db_user.id, "username":
db_user.username}

@app.get("/api/me")
def get_me(token: str, db: Session = Depends(get_db)):
    if token not in sessions:
        raise HTTPException(status_code=401, detail="Не
авторизован")
```

```
    user_id = sessions[token]
    user = db.query(User).filter(User.id == user_id).first()
    return {"id": user.id, "username": user.username}

@app.get("/api/products", response_model=List[ProductResponse])
def get_products(category: str = None, db: Session =
Depends(get_db)):
    query = db.query(Product)
    if category:
        query = query.filter(Product.category == category)
    return query.all()

@app.get("/api/products/{product_id}",
response_model=ProductResponse)
def get_product(product_id: int, db: Session = Depends(get_db)):
    product = db.query(Product).filter(Product.id ==
product_id).first()
    if not product:
        raise HTTPException(status_code=404, detail="Товар не
найден")
    return product

@app.post("/api/events")
def add_event(event: EventCreate, token: str, db: Session =
Depends(get_db)):
    if token not in sessions:
        raise HTTPException(status_code=401, detail="Не
авторизован")

    user_id = sessions[token]

    new_event = Event(
        user_id=user_id,
        product_id=event.product_id,
        event_type=event.event_type,
        duration_seconds=event.duration_seconds
    )
    db.add(new_event)
    db.commit()

    return {"message": "Событие сохранено"}

@app.get("/api/cart")
def get_cart(token: str, db: Session = Depends(get_db)):
    if token not in sessions:
        raise HTTPException(status_code=401, detail="Не
авторизован")

    user_id = sessions[token]
    cart_items = db.query(Cart).filter(Cart.user_id ==
user_id).all()
```

```
    result = []
    for item in cart_items:
        product = db.query(Product).filter(Product.id ==
item.product_id).first()
        if product:
            result.append({
                "product_id": product.id,
                "name": product.name,
                "price": product.price,
                "quantity": item.quantity
            })

    return result

@app.post("/api/cart")
def add_to_cart(item: CartItem, token: str, db: Session =
Depends(get_db)):
    if token not in sessions:
        raise HTTPException(status_code=401, detail="Не
авторизован")

    user_id = sessions[token]

    existing = db.query(Cart).filter(Cart.user_id == user_id,
Cart.product_id == item.product_id).first()
    if existing:
        existing.quantity += item.quantity
    else:
        new_item = Cart(user_id=user_id, product_id=item.product_id,
quantity=item.quantity)
        db.add(new_item)

    db.commit()
    return {"message": "Товар добавлен в корзину"}

@app.delete("/api/cart/{product_id}")
def remove_from_cart(product_id: int, token: str, db: Session =
Depends(get_db)):
    if token not in sessions:
        raise HTTPException(status_code=401, detail="Не
авторизован")

    user_id = sessions[token]
    item = db.query(Cart).filter(Cart.user_id == user_id,
Cart.product_id == product_id).first()
    if item:
        db.delete(item)
        db.commit()

    return {"message": "Товар удален из корзины"}
```

```
@app.get("/api/recommendations",
response_model=List[RecommendationResponse])
def get_recommendations(token: str, n: int = 6, db: Session =
Depends(get_db)):
    if token not in sessions:
        raise HTTPException(status_code=401, detail="Не
авторизован")

    user_id = sessions[token]

    rec_ids = recommender.get_recommendations(user_id, n)

    result = []
    for idx, prod_id in enumerate(rec_ids):
        product = db.query(Product).filter(Product.id ==
prod_id).first()
        if product:
            result.append({
                "product_id": product.id,
                "name": product.name,
                "price": product.price,
                "image_url": product.image_url,
                "score": 1.0 - idx * 0.1
            })

    return result

@app.get("/api/similar/{product_id}",
response_model=List[RecommendationResponse])
def get_similar(product_id: int, n: int = 4, db: Session =
Depends(get_db)):
    similar_ids = recommender.get_similar_products(product_id, n)

    result = []
    for prod_id in similar_ids:
        product = db.query(Product).filter(Product.id ==
prod_id).first()
        if product:
            result.append({
                "product_id": product.id,
                "name": product.name,
                "price": product.price,
                "image_url": product.image_url,
                "score": 0.8
            })

    return result

@app.get("/product/{product_id}", response_class=HTMLResponse)
async def product_page(request: Request, product_id: int, db:
Session = Depends(get_db)):
```

```
    product = db.query(Product).filter(Product.id ==
product_id).first()
    if not product:
        return HTMLResponse(content="<h1>Товар не найден</h1>",
status_code=404)

    similar_ids = recommender.get_similar_products(product_id, 4)
    similar_products = []
    for pid in similar_ids:
        p = db.query(Product).filter(Product.id == pid).first()
        if p:
            similar_products.append(p)

    return templates.TemplateResponse("product.html", {
        "request": request,
        "product": product,
        "similar_products": similar_products
    })

@app.get("/", response_class=HTMLResponse)
def index():
    with open("static/index.html", "r", encoding="utf-8") as f:
        return f.read()
```

Файл «models.py»

```
from pydantic import BaseModel
from typing import List, Optional

class UserCreate(BaseModel):
    username: str
    password: str

class UserLogin(BaseModel):
    username: str
    password: str

class ProductResponse(BaseModel):
    id: int
    name: str
    category: str
    price: float
    brand: str
    description: str
    image_url: str

class EventCreate(BaseModel):
    product_id: int
    event_type: str
    duration_seconds: int = 0

class CartItem(BaseModel):
```

```
    product_id: int
    quantity: int

class RecommendationResponse(BaseModel):
    product_id: int
    name: str
    price: float
    image_url: str
    score: float
```

Файл «recommendations.py»

```
import pandas as pd
import numpy as np
from scipy.sparse import csr_matrix
from sklearn.metrics.pairwise import cosine_similarity
from database import SessionLocal, Event, Product

class RecommenderSystem:
    def __init__(self):
        self.user_item_matrix = None
        self.user_ids = []
        self.product_ids = []
        self.user_to_idx = {}
        self.product_to_idx = {}
        self.similarity_matrix = None
        self.all_product_ids = []

    def build_matrix(self):
        db = SessionLocal()
        events = db.query(Event).all()
        products = db.query(Product).all()
        db.close()

        self.all_product_ids = [p.id for p in products]

        if not events:
            return None

        data = []
        for event in events:
            weight = 1 if event.event_type == 'view' else (3 if
event.event_type == 'cart' else 10)
            data.append({
                'user_id': event.user_id,
                'product_id': event.product_id,
                'weight': weight
            })

        df = pd.DataFrame(data)

        self.user_ids = df['user_id'].unique()
```

```
        self.product_ids = df['product_id'].unique()
        self.user_to_idx = {uid: i for i, uid in
enumerate(self.user_ids)}
        self.product_to_idx = {pid: i for i, pid in
enumerate(self.product_ids)}

        rows = [self.user_to_idx[uid] for uid in df['user_id']]
        cols = [self.product_to_idx[pid] for pid in
df['product_id']]
        data = df['weight'].values

        self.user_item_matrix = csr_matrix((data, (rows, cols)),
                                           shape=(len(self.user_ids)
, len(self.product_ids)))
        return self.user_item_matrix

    def train(self):
        matrix = self.build_matrix()
        if matrix is None:
            print("Нет данных для обучения.")
            return False

        item_matrix = matrix.T
        self.similarity_matrix = cosine_similarity(item_matrix)
        print(f"Модель обучена. Пользователей: {len(self.user_ids)},
Товаров: {len(self.product_ids)}")
        return True

    def get_user_viewed_products(self, user_id: int):
        """Возвращает множество ID товаров, которые пользователь уже
видел"""
        db = SessionLocal()
        events = db.query(Event.product_id).filter(Event.user_id ==
user_id).all()
        db.close()
        return set(e[0] for e in events)

    def get_recommendations(self, user_id: int, n: int = 6):
        """Возвращает рекомендации, исключая просмотренные товары"""


        all_products = set(self.all_product_ids)
        viewed = self.get_user_viewed_products(user_id)

        unseen_products = list(all_products - viewed)

        if not unseen_products:
            return self.get_popular_products(n, viewed)


        if user_id not in self.user_to_idx:
            return self.get_popular_products(n, viewed)
```

```
        user_idx = self.user_to_idx[user_id]
        user_vector =
self.user_item_matrix[user_idx].toarray().flatten()

        if np.sum(user_vector) == 0:
            return self.get_popular_products(n, viewed)

        top_items = np.argsort(user_vector)[-5:][::-1]

        candidates = {}
        for item_idx in top_items:
            if self.product_ids[item_idx] in viewed:
                continue
            similar = np.argsort(self.similarity_matrix[item_idx])[-
n*4:][::-1]
            for idx in similar:
                product_id = self.product_ids[idx]
                if product_id in viewed:
                    continue
                if idx < len(self.product_ids):
                    candidates[product_id] =
candidates.get(product_id, 0) +
self.similarity_matrix[item_idx][idx]

        sorted_candidates = sorted(candidates.items(), key=lambda x:
x[1], reverse=True)
        recommended = [pid for pid, _ in sorted_candidates[:n]]
        if len(recommended) < n:
            popular = self.get_popular_products(n*2, viewed)
            for pid in popular:
                if pid not in recommended:
                    recommended.append(pid)
                    if len(recommended) >= n:
                        break

        return recommended[:n]

    def get_popular_products(self, n: int = 6, exclude: set = None):
        """Возвращает популярные товары, исключая просмотренные"""
        if exclude is None:
            exclude = set()

        db = SessionLocal()
        events = db.query(Event).all()
        db.close()

        if not events:
            db = SessionLocal()
            products =
db.query(Product).filter(Product.id.notin_(list(exclude))).limit(n).
all()
```

```
            db.close()
            return [p.id for p in products]

        popularity = {}
        for event in events:
            if event.product_id in exclude:
                continue
            weight = 1 if event.event_type == 'view' else (3 if
event.event_type == 'cart' else 10)
            popularity[event.product_id] =
popularity.get(event.product_id, 0) + weight

        if not popularity:
            db = SessionLocal()
            products =
db.query(Product).filter(Product.id.notin_(list(exclude))).limit(n).
all()
            db.close()
            return [p.id for p in products]

        sorted_products = sorted(popularity.items(), key=lambda x:
x[1], reverse=True)
        result = [pid for pid, _ in sorted_products[:n]]


        if len(result) < n:
            db = SessionLocal()
            existing_ids = [p[0] for p in sorted_products]
            additional = db.query(Product.id).filter(
                Product.id.notin_(list(exclude) + existing_ids)
            ).limit(n - len(result)).all()
            db.close()
            result.extend([p[0] for p in additional])

        return result

    def get_similar_products(self, product_id: int, n: int = 4):
        if self.similarity_matrix is None:
            return []

        if product_id not in self.product_to_idx:
            return []

        product_idx = self.product_to_idx[product_id]
        similar_scores =
list(enumerate(self.similarity_matrix[product_idx]))
        similar_scores.sort(key=lambda x: x[1], reverse=True)

        result = []
        for idx, score in similar_scores:
            if idx != product_idx and idx < len(self.product_ids):
                result.append(self.product_ids[idx])
```

```
                if len(result) >= n:
                    break
        return result

recommender = RecommenderSystem()
```

Файл «add_events»

```
import sqlite3
from datetime import datetime
import random

conn = sqlite3.connect('shop.db')
cursor = conn.cursor()

cursor.execute("SELECT id FROM users LIMIT 1")
user = cursor.fetchone()

if not user:
    print("Нет пользователей. Зарегистрируйтесь в системе.")
    exit()

user_id = user[0]
print(f"Пользователь ID: {user_id}")

cursor.execute("SELECT id FROM products")
products = [row[0] for row in cursor.fetchall()]

if not products:
    print("Нет товаров в базе данных.")
    exit()

print(f"Найдено {len(products)} товаров")

viewed = random.sample(products, min(10, len(products)))

for product_id in viewed:
    duration = random.randint(10, 60)
    cursor.execute('''
        INSERT INTO events (user_id, product_id, event_type,
duration_seconds, timestamp)
        VALUES (?, ?, 'view', ?, ?)
    ''', (user_id, product_id, duration, datetime.now()))
    print(f"Просмотр товара {product_id} ({duration} сек)")

cart_items = random.sample(viewed, min(3, len(viewed)))
for product_id in cart_items:
    cursor.execute('''
        INSERT INTO events (user_id, product_id, event_type,
duration_seconds, timestamp)
        VALUES (?, ?, 'cart', 0, ?)
    ''', (user_id, product_id, datetime.now()))
```

```
    print(f"Добавление в корзину товара {product_id}")

conn.commit()
conn.close()

print(f"Добавлено {len(viewed)} просмотров и {len(cart_items)}
корзин для пользователя {user_id}")
```

Файл «database.py»

```
from sqlalchemy import create_engine, Column, Integer, String,
Float, ForeignKey, DateTime
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker, relationship
from datetime import datetime

engine = create_engine('sqlite:///shop.db',
connect_args={'check_same_thread': False})
Base = declarative_base()
SessionLocal = sessionmaker(bind=engine)

class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True, index=True)
    username = Column(String, unique=True, index=True)
    password = Column(String)

class Product(Base):
    __tablename__ = 'products'
    id = Column(Integer, primary_key=True, index=True)
    name = Column(String)
    category = Column(String)
    price = Column(Float)
    brand = Column(String)
    description = Column(String)
    image_url = Column(String)

class Event(Base):
    __tablename__ = 'events'
    id = Column(Integer, primary_key=True, index=True)
    user_id = Column(Integer, ForeignKey('users.id'))
    product_id = Column(Integer, ForeignKey('products.id'))
    event_type = Column(String)  # view, cart, purchase
    duration_seconds = Column(Integer, default=0)
    timestamp = Column(DateTime, default=datetime.now)

class Cart(Base):
    __tablename__ = 'cart'
    id = Column(Integer, primary_key=True, index=True)
    user_id = Column(Integer, ForeignKey('users.id'))
    product_id = Column(Integer, ForeignKey('products.id'))
    quantity = Column(Integer, default=1)
```

```
Base.metadata.create_all(bind=engine)

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

Файл «static/index.html»

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>ElectroShop</title>
    <link rel="stylesheet" href="/static/style.css">
    <style>
        /* Модальные окна — минимальные стили, чтобы точно работали
*/
        .modal-overlay {
            display: none;
            position: fixed;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
            background: rgba(0,0,0,0.5);
            justify-content: center;
            align-items: center;
            z-index: 9999;
        }
        .modal-overlay.active {
            display: flex;
        }
        .modal-window {
            background: white;
            padding: 35px 30px;
            border-radius: 16px;
            max-width: 380px;
            width: 90%;
            box-shadow: 0 20px 60px rgba(0,0,0,0.3);
            position: relative;
        }
        .modal-window .close {
            position: absolute;
            top: 12px;
            right: 18px;
            font-size: 24px;
```

```
        cursor: pointer;
        color: #999;
        background: none;
        border: none;
    }
    .modal-window .close:hover {
        color: #333;
    }
    .modal-window h2 {
        text-align: center;
        margin-bottom: 5px;
        font-size: 24px;
        color: #333;
    }
    .modal-window .sub {
        text-align: center;
        color: #888;
        font-size: 14px;
        margin-bottom: 20px;
    }
    .modal-window label {
        display: block;
        font-size: 14px;
        font-weight: 600;
        color: #444;
        margin-top: 12px;
        margin-bottom: 4px;
    }
    .modal-window input {
        width: 100%;
        padding: 10px 14px;
        border: 2px solid #e0e0e0;
        border-radius: 8px;
        font-size: 15px;
        box-sizing: border-box;
    }
    .modal-window input:focus {
        border-color: #667eea;
        outline: none;
    }
    .modal-window .btn {
        width: 100%;
        padding: 12px;
        border: none;
        border-radius: 8px;
        font-size: 16px;
        font-weight: 600;
        cursor: pointer;
        margin-top: 18px;
        color: white;
    }
    .modal-window .btn-login {
```

```
            background: linear-gradient(135deg, #667eea, #764ba2);
        }
        .modal-window .btn-register {
            background: linear-gradient(135deg, #28a745, #20c997);
        }
        .modal-window .switch {
            text-align: center;
            margin-top: 14px;
            font-size: 14px;
            color: #888;
        }
        .modal-window .switch a {
            color: #667eea;
            font-weight: 600;
            cursor: pointer;
        }
        .modal-window .error {
            background: #fee2e2;
            color: #dc2626;
            padding: 8px 12px;
            border-radius: 6px;
            font-size: 14px;
            display: none;
            text-align: center;
            margin-bottom: 10px;
        }
        .modal-window .error.show {
            display: block;
        }
    </style>
</head>
<body>
    <header>
        <h1 onclick="location.href='/'">📱 ElectroShop</h1>
        <div id="auth-section">
            <button onclick="openLogin()" class="login-
btn">Войти</button>
            <button onclick="openRegister()" class="register-
btn">Регистрация</button>
            <div id="user-info" style="display:none; align-
items:center; gap:10px; color:white; font-weight:500;">
                <span id="username">Пользователь</span>
                <button onclick="logout()" class="logout-
btn">Выйти</button>
            </div>
        </div>
    </header>

    <!-- МОДАЛКА ВХОДА -->
    <div class="modal-overlay" id="loginModal">
        <div class="modal-window">
            <button class="close" onclick="closeLogin()">×</button>
```

```
            <h2>Вход</h2>
            <p class="sub">Войдите в аккаунт</p>
            <div class="error" id="loginError">Неверный логин или
пароль</div>
            <label>Логин</label>
            <input type="text" id="loginUser" placeholder="Введите
логин">
            <label>Пароль</label>
            <input type="password" id="loginPass"
placeholder="Введите пароль">
            <button class="btn btn-login"
onclick="doLogin()">Войти</button>
            <div class="switch">Нет аккаунта? <a
onclick="switchToRegister()">Зарегистрироваться</a></div>
        </div>
    </div>

    <!-- МОДАЛКА РЕГИСТРАЦИИ -->
    <div class="modal-overlay" id="registerModal">
        <div class="modal-window">
            <button class="close"
onclick="closeRegister()">×</button>
            <h2>Регистрация</h2>
            <p class="sub">Создайте аккаунт</p>
            <div class="error" id="registerError">Пользователь уже
существует</div>
            <label>Логин</label>
            <input type="text" id="regUser" placeholder="Придумайте
логин">
            <label>Пароль</label>
            <input type="password" id="regPass"
placeholder="Придумайте пароль">
            <button class="btn btn-register"
onclick="doRegister()">Зарегистрироваться</button>
            <div class="switch">Уже есть аккаунт? <a
onclick="switchToLogin()">Войти</a></div>
        </div>
    </div>

    <main>
        <div class="sidebar">
            <h3>Поиск</h3>
            <input type="text" id="search-input"
placeholder="Название или бренд...">

            <h3>Категории</h3>
            <button onclick="setCategory('all')">Все товары</button>
            <button
onclick="setCategory('smartphone')">Смартфоны</button>
            <button
onclick="setCategory('laptop')">Ноутбуки</button>
```

```
            <button
onclick="setCategory('accessory')">Аксессуары</button>
            <button
onclick="setCategory('tablet')">Планшеты</button>
            <button onclick="setCategory('gaming')">Игры</button>

            <h3>Цена</h3>
            <div class="price-inputs">
                <input type="number" id="price-min" placeholder="от
₽">
                <input type="number" id="price-max" placeholder="до
₽">
            </div>
            <button onclick="applyPriceFilter()"
style="background:#28a745; color:white; border:none; padding:8px;
border-radius:5px; cursor:pointer; width:100%;">Применить
цену</button>

            <h3>Бренды</h3>
            <div id="brands-list"></div>
            <button onclick="clearBrandFilter()"
style="background:#6c757d; color:white; border:none; padding:8px;
border-radius:5px; cursor:pointer; width:100%; margin-
top:5px;">Сбросить бренды</button>

            <h3>Рекомендации</h3>
            <button onclick="loadRecommendations()"
style="background:#764ba2; color:white; border:none; padding:8px;
border-radius:5px; cursor:pointer; width:100%;">Для вас</button>
        </div>

        <div class="content">
            <div id="recommendations-banner" class="recommendations-
banner" style="display:none;">
                <div style="display:flex; justify-content:space-
between; align-items:center; margin-bottom:15px;">
                    <h2 style="margin:0;">Рекомендуем для вас</h2>
                    <button
onclick="document.getElementById('recommendations-
banner').style.display='none'" style="background:none; border:none;
font-size:28px; cursor:pointer; color:#888; padding:0
10px;">✕</button>
                </div>
                <div id="recommendations-list" class="products-
grid"></div>
            </div>
            <div id="products-section">
                <h2 id="products-title">Все товары</h2>
                <div id="filter-badges"></div>
                <div id="products-list" class="products-grid"></div>
            </div>
        </div>
```

```
        <div class="cart-sidebar">
            <h3>Корзина</h3>
            <div id="cart-items"></div>
            <div id="cart-total"></div>
        </div>
    </main>

    <script src="/static/script.js"></script>
</body>
</html>
```

Файл «static/product.html»

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>{{ product.name }} | ElectroShop</title>
    <link rel="stylesheet" href="/static/style.css">
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: -apple-system, BlinkMacSystemFont, 'Segoe
UI', Roboto, sans-serif;
            background: #f5f5f5;
        }

        header {
            background: linear-gradient(135deg, #667eea 0%, #764ba2
100%);
            color: white;
            padding: 1rem 2rem;
            display: flex;
            justify-content: space-between;
            align-items: center;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }

        header h1 {
            font-size: 24px;
            cursor: pointer;
        }

        #auth-section input, #auth-section button {
```

```
        padding: 8px 12px;
        margin: 0 5px;
        border: none;
        border-radius: 5px;
    }

    #auth-section button {
        background: white;
        color: #667eea;
        cursor: pointer;
        font-weight: bold;
    }

    main {
        max-width: 1200px;
        margin: 20px auto;
        padding: 0 20px;
    }

    /* КАРТОЧКА ТОВАРА - ОСНОВНОЙ БЛОК */
    .product-detail {
        display: flex;
        gap: 40px;
        margin-bottom: 40px;
        background: white;
        padding: 30px;
        border-radius: 15px;
        box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        flex-wrap: wrap;
    }

    /* БЛОК С ФОТО - ФИКСИРОВАННЫЙ РАЗМЕР */
    .product-image {
        flex: 1;
        min-width: 280px;
        text-align: center;
    }

    .image-container {
        width: 100%;
        max-width: 350px;
        height: 350px;
        margin: 0 auto;
        display: flex;
        align-items: center;
        justify-content: center;
        background: #f8f9fa;
        border-radius: 15px;
        overflow: hidden;
    }

    .image-container img {
```

```
        width: 100%;
        height: 100%;
        object-fit: contain;
        transition: transform 0.3s;
    }

    .image-container img:hover {
        transform: scale(1.05);
    }

    /* БЛОК С ИНФОРМАЦИЕЙ */
    .product-info {
        flex: 2;
        min-width: 280px;
    }

    .product-info h1 {
        font-size: 28px;
        margin-bottom: 10px;
        color: #333;
    }

    .product-brand {
        color: #667eea;
        font-size: 18px;
        margin-bottom: 15px;
        font-weight: bold;
    }

    .product-price {
        font-size: 32px;
        font-weight: bold;
        color: #667eea;
        margin: 20px 0;
    }

    .product-description {
        font-size: 16px;
        line-height: 1.6;
        color: #555;
        margin: 20px 0;
        padding: 15px;
        background: #f8f9fa;
        border-radius: 10px;
    }

    .product-specs {
        margin: 20px 0;
        padding: 15px;
        background: #f8f9fa;
        border-radius: 10px;
    }
```

```
.product-specs h3 {
    margin-bottom: 10px;
    color: #333;
}

.product-specs ul {
    list-style: none;
    padding: 0;
}

.product-specs li {
    padding: 5px 0;
    border-bottom: 1px solid #eee;
}

.add-to-cart-btn {
    background: #667eea;
    color: white;
    border: none;
    padding: 15px 30px;
    font-size: 18px;
    border-radius: 8px;
    cursor: pointer;
    margin-right: 15px;
    transition: background 0.3s;
}

.add-to-cart-btn:hover {
    background: #5a67d8;
}

.back-btn {
    background: #6c757d;
    color: white;
    border: none;
    padding: 15px 30px;
    font-size: 18px;
    border-radius: 8px;
    cursor: pointer;
    text-decoration: none;
    display: inline-block;
}

.back-btn:hover {
    background: #5a6268;
}

.timer-info {
    margin-top: 20px;
    font-size: 12px;
    color: #888;
```

```
        }

        /* ПОХОЖИЕ ТОВАРЫ */
        .similar-products {
            margin-top: 40px;
        }

        .similar-products h2 {
            margin-bottom: 20px;
            color: #333;
        }

        .similar-grid {
            display: grid;
            grid-template-columns: repeat(auto-fill, minmax(200px,
1fr));
            gap: 20px;
        }

        .similar-card {
            background: white;
            border-radius: 10px;
            padding: 15px;
            text-align: center;
            box-shadow: 0 2px 5px rgba(0,0,0,0.1);
            transition: transform 0.2s;
            cursor: pointer;
        }

        .similar-card:hover {
            transform: translateY(-5px);
            box-shadow: 0 5px 20px rgba(0,0,0,0.15);
        }

        .similar-card img {
            width: 100%;
            height: 150px;
            object-fit: contain;
            margin-bottom: 10px;
        }

        .similar-card h3 {
            font-size: 14px;
            margin: 10px 0;
        }

        .similar-card .price {
            font-weight: bold;
            color: #667eea;
        }

        /* МОДАЛЬНОЕ ОКНО */
```

```
        .modal {
            display: none;
            position: fixed;
            top: 50%;
            left: 50%;
            transform: translate(-50%, -50%);
            background: white;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 10px 40px rgba(0,0,0,0.2);
            z-index: 1000;
        }

        .modal input {
            width: 100%;
            padding: 8px;
            margin: 10px 0;
            border: 1px solid #ddd;
            border-radius: 5px;
        }
    </style>
</head>
<body>
    <header>
        <h1 onclick="location.href='/'">ElectroShop</h1>
        <div id="auth-section">
            <div id="login-form">
                <input type="text" id="login-username"
placeholder="Логин">
                <input type="password" id="login-password"
placeholder="Пароль">
                <button onclick="login()">Вход</button>
                <button
onclick="showRegister()">Регистрация</button>
            </div>
            <div id="user-info" style="display:none">
                <span id="username">Пользователь</span>
                <button onclick="logout()">Выход</button>
            </div>
        </div>
    </header>

    <div id="register-modal" class="modal">
        <h3>Регистрация</h3>
        <input type="text" id="reg-username" placeholder="Логин">
        <input type="password" id="reg-password"
placeholder="Пароль">
        <button onclick="register()" style="background:#28a745;
color:white;">Зарегистрироваться</button>
        <button onclick="hideRegister()">Отмена</button>
    </div>
```

```
    <main>
        <div class="product-detail">
            <!-- ФОТО ТОВАРА - ФИКСИРОВАННЫЙ РАЗМЕР 350x350 -->
            <div class="product-image">
                <div class="image-container">
                    <img src="{{ product.image_url }}"
alt="{{ product.name }}"
                         onerror="this.src='https://via.placeholder.
com/350x350?text=+Нет+фото'">
                </div>
            </div>

            <div class="product-info">
                <h1>{{ product.name }}</h1>
                <div class="product-brand">{{ product.brand }}</div>
                <div class="product-
price">{{ "{:,.0f}".format(product.price) }} ₽</div>

                <div class="product-description">
                    <strong>Описание:</strong><br>
                    {{ product.description }}
                </div>

                <div class="product-specs">
                    <h3>Характеристики</h3>
                    <ul>
                        <li>Категория: {{ product.category }}</li>
                        <li>Бренд: {{ product.brand }}</li>
                        <li>Артикул: #{{ product.id }}</li>
                        <li>Гарантия: 12 месяцев</li>
                        <li>Доставка: 1-3 дня</li>
                    </ul>
                </div>

                <button class="add-to-cart-btn"
onclick="addToCart({{ product.id }})"> Добавить в корзину</button>
                <button class="back-btn"
onclick="location.href='/'">← Назад в каталог</button>

                <div class="timer-info" id="view-timer">
                    ⏱ Вы изучаете товар: <span
id="seconds">0</span> сек.
                </div>
            </div>
        </div>

        {% if similar_products %}
        <div class="similar-products">
            <h2>Похожие товары</h2>
            <div class="similar-grid">
                {% for p in similar_products %}
```

```
                <div class="similar-card"
onclick="location.href='/product/{{ p.id }}'">
                    <img src="{{ p.image_url }}" alt="{{ p.name }}"
                         onerror="this.src='https://via.placeholder.
com/150x150?text='">
                    <h3>{{ p.name }}</h3>
                    <div
class="price">{{ "{:,.0f}".format(p.price) }} ₽</div>
                </div>
                {% endfor %}
            </div>
        </div>
        {% endif %}
    </main>

    <script>
        let token = localStorage.getItem('token');
        let startTime = Date.now();
        let timerInterval;

        function startTimer() {
            timerInterval = setInterval(() => {
                if (token) {
                    let seconds = Math.floor((Date.now() -
startTime) / 1000);
                    document.getElementById('seconds').textContent =
seconds;
                }
            }, 1000);
        }

        function sendViewEvent() {
            if (token) {
                let duration = Math.floor((Date.now() - startTime) /
1000);
                fetch(`/api/events?token=${token}`, {
                    method: 'POST',
                    headers: {'Content-Type': 'application/json'},
                    body: JSON.stringify({
                        product_id: {{ product.id }},
                        event_type: 'view',
                        duration_seconds: duration
                    })
                });
            }
        }

        startTimer();
        window.addEventListener('beforeunload', sendViewEvent);

        if (token) {
```

```
            document.getElementById('login-form').style.display =
'none';
            document.getElementById('user-info').style.display =
'block';
        }

        async function addToCart(productId) {
            if (!token) {
                alert('Войдите, чтобы добавить товар в корзину');
                return;
            }

            await fetch(`/api/events?token=${token}`, {
                method: 'POST',
                headers: {'Content-Type': 'application/json'},
                body: JSON.stringify({product_id: productId,
event_type: 'cart', duration_seconds: 0})
            });

            await fetch(`/api/cart?token=${token}`, {
                method: 'POST',
                headers: {'Content-Type': 'application/json'},
                body: JSON.stringify({product_id: productId,
quantity: 1})
            });

            alert('Товар добавлен в корзину!');
        }

        async function login() {
            const username = document.getElementById('login-
username').value;
            const password = document.getElementById('login-
password').value;

            const response = await fetch('/api/login', {
                method: 'POST',
                headers: {'Content-Type': 'application/json'},
                body: JSON.stringify({username, password})
            });

            if (response.ok) {
                const data = await response.json();
                token = data.token;
                localStorage.setItem('token', token);
                location.reload();
            } else {
                alert('Ошибка входа');
            }
        }

        async function register() {
```

```
            const username = document.getElementById('reg-
username').value;
            const password = document.getElementById('reg-
password').value;

            const response = await fetch('/api/register', {
                method: 'POST',
                headers: {'Content-Type': 'application/json'},
                body: JSON.stringify({username, password})
            });

            if (response.ok) {
                alert('Регистрация успешна! Теперь войдите.');
                hideRegister();
            } else {
                alert('Пользователь уже существует');
            }
        }

        function logout() {
            localStorage.removeItem('token');
            token = null;
            location.href = '/';
        }

        function showRegister() {
            document.getElementById('register-modal').style.display
= 'block';
        }

        function hideRegister() {
            document.getElementById('register-modal').style.display
= 'none';
        }
    </script>
</body>
</html>
```

Файл «static/script.js»

```
let token = localStorage.getItem('token');
let allProducts = [];
let selectedBrands = new Set();
let currentCategory = 'all';
let searchQuery = '';
let priceMin = null;
let priceMax = null;

document.addEventListener('DOMContentLoaded', () => {
    if (token) {
        showLoggedInUI();
        loadCart();
```

```
    }
    loadAllProducts();
    document.getElementById('search-
input').addEventListener('input', (e) => {
        searchQuery = e.target.value.toLowerCase();
        applyFilters();
    });
});

async function loadAllProducts() {
    const response = await fetch('/api/products');
    allProducts = await response.json();
    loadBrandsList();
    applyFilters();
}

function loadBrandsList() {
    const brands = [...new Set(allProducts.map(p =>
p.brand))].sort();
    document.getElementById('brands-list').innerHTML = brands.map(b
=>
        `<label><input type="checkbox" value="${b}"
onchange="toggleBrand(this.value)"> ${b}</label>`
    ).join('');
}

function toggleBrand(brand) {
    selectedBrands.has(brand) ? selectedBrands.delete(brand) :
selectedBrands.add(brand);
    applyFilters();
}

function clearBrandFilter() {
    selectedBrands.clear();
    document.querySelectorAll('#brands-list input').forEach(cb =>
cb.checked = false);
    applyFilters();
}

function applyPriceFilter() {
    priceMin = parseInt(document.getElementById('price-min').value)
|| null;
    priceMax = parseInt(document.getElementById('price-max').value)
|| null;
    applyFilters();
}

function setCategory(cat) {
    currentCategory = cat;
    applyFilters();
}
```

```
function applyFilters() {
    let filtered = [...allProducts];
    if (currentCategory !== 'all') filtered = filtered.filter(p =>
p.category === currentCategory);
    if (searchQuery) filtered = filtered.filter(p =>
p.name.toLowerCase().includes(searchQuery) ||
p.brand.toLowerCase().includes(searchQuery));
    if (selectedBrands.size) filtered = filtered.filter(p =>
selectedBrands.has(p.brand));
    if (priceMin) filtered = filtered.filter(p => p.price >=
priceMin);
    if (priceMax) filtered = filtered.filter(p => p.price <=
priceMax);
    displayProducts(filtered);
    updateFilterBadges();
}

function displayProducts(products) {
    const container = document.getElementById('products-list');
    document.getElementById('products-title').textContent = `Товары
(${products.length})`;
    if (!products.length) {
        container.innerHTML = '<div class="empty-state">Товаров не
найдено</div>';
        return;
    }
    container.innerHTML = products.map(p => `
        <div class="product-card"
onclick="location.href='/product/${p.id}'">
            <h3>${p.name}</h3>
            <div class="brand">${p.brand}</div>
            <div class="price">${p.price.toLocaleString()} ₽</div>
            <button onclick="event.stopPropagation();
addToCart(${p.id})">В корзину</button>
            <button onclick="event.stopPropagation();
viewSimilar(${p.id})">Похожие</button>
        </div>
    `).join('');
}

function updateFilterBadges() {
    const container = document.getElementById('filter-badges');
    let active = [];
    if (currentCategory !== 'all') active.push(`Категория:
${currentCategory}`);
    if (searchQuery) active.push(`Поиск: "${searchQuery}"`);
    if (selectedBrands.size) active.push(`Бренды:
${[...selectedBrands].join(', ')}`);
    if (priceMin) active.push(`Цена от ${priceMin}₽`);
    if (priceMax) active.push(`Цена до ${priceMax}₽`);
    container.innerHTML = active.length ? `
        <div style="margin:10px 0;">
```

```
            ${active.map(f => `<span class="filter-
badge">${f}</span>`).join(' ')}
            <button class="clear-all-btn"
onclick="clearAllFilters()">Очистить всё</button>
        </div>
    ` : '';
}

function clearAllFilters() {
    currentCategory = 'all';
    searchQuery = '';
    selectedBrands.clear();
    priceMin = null;
    priceMax = null;
    document.getElementById('search-input').value = '';
    document.getElementById('price-min').value = '';
    document.getElementById('price-max').value = '';
    document.querySelectorAll('#brands-list input').forEach(cb =>
cb.checked = false);
    applyFilters();
}

async function loadRecommendations() {
    if (!token) { alert('Войдите, чтобы видеть рекомендации');
return; }
    const res = await fetch(`/api/recommendations?token=${token}`);
    const recs = await res.json();
    const banner = document.getElementById('recommendations-
banner');
    const container = document.getElementById('recommendations-
list');
    if (recs.length) {
        container.innerHTML = recs.map(r => `
            <div class="product-card"
onclick="location.href='/product/${r.product_id}'">
                <h3>${r.name}</h3>
                <div class="price">${r.price.toLocaleString()}
₽</div>
                <button onclick="event.stopPropagation();
addToCart(${r.product_id})">В корзину</button>
            </div>
        `).join('');
        banner.style.display = 'block';
        banner.scrollIntoView({ behavior: 'smooth' });
    } else {
        alert('Посмотрите несколько товаров, и рекомендации
появятся!');
    }
}

async function viewSimilar(id) {
    const res = await fetch(`/api/similar/${id}`);
```

```
    const data = await res.json();
    alert(data.length ? 'Похожие:\n' + data.map(s =>
s.name).join('\n') : 'Похожих нет');
}

async function addToCart(id) {
    if (!token) { alert('Войдите, чтобы добавить в корзину');
return; }
    await fetch(`/api/events?token=${token}`, { method: 'POST',
headers: {'Content-Type':'application/json'}, body:
JSON.stringify({product_id:id, event_type:'cart',
duration_seconds:0}) });
    await fetch(`/api/cart?token=${token}`, { method: 'POST',
headers: {'Content-Type':'application/json'}, body:
JSON.stringify({product_id:id, quantity:1}) });
    loadCart();
}

async function loadCart() {
    if (!token) return;
    const res = await fetch(`/api/cart?token=${token}`);
    const cart = await res.json();
    const container = document.getElementById('cart-items');
    let total = 0;
    if (!cart.length) {
        container.innerHTML = '<div style="padding:20px;text-
align:center;color:#888;">Корзина пуста</div>';
        document.getElementById('cart-total').innerHTML = '';
        return;
    }
    container.innerHTML = cart.map(item => {
        total += item.price * item.quantity;
        return `<div class="cart-
item"><span>${item.name}<br><small>x${item.quantity}</small></span><
span>${(item.price*item.quantity).toLocaleString()} ₽</span><button
onclick="removeFromCart(${item.product_id})">×</button></div>`;
    }).join('');
    document.getElementById('cart-total').innerHTML = `Итого:
${total.toLocaleString()} ₽`;
}

async function removeFromCart(id) {
    await fetch(`/api/cart/${id}?token=${token}`, { method:
'DELETE' });
    loadCart();
}

function showLoggedInUI() {
    document.querySelector('.login-btn').style.display = 'none';
    document.querySelector('.register-btn').style.display = 'none';
    document.getElementById('user-info').style.display = 'flex';
}
```

```
function showLoggedOutUI() {
    document.querySelector('.login-btn').style.display = 'inline-
block';
    document.querySelector('.register-btn').style.display = 'inline-
block';
    document.getElementById('user-info').style.display = 'none';
}

function logout() {
    localStorage.removeItem('token');
    token = null;
    showLoggedOutUI();
    document.getElementById('cart-items').innerHTML = '';
    document.getElementById('cart-total').innerHTML = '';
    applyFilters();
}

// ===== МОДАЛКИ =====
function openLogin() {
    document.getElementById('loginModal').classList.add('active');
    document.getElementById('loginError').classList.remove('show');
    document.getElementById('loginUser').value = '';
    document.getElementById('loginPass').value = '';
}

function closeLogin() {
    document.getElementById('loginModal').classList.remove('active')
;
}

function openRegister() {
    document.getElementById('registerModal').classList.add('active')
;
    document.getElementById('registerError').classList.remove('show'
);
    document.getElementById('regUser').value = '';
    document.getElementById('regPass').value = '';
}

function closeRegister() {
    document.getElementById('registerModal').classList.remove('activ
e');
}

function switchToRegister() {
    closeLogin();
    setTimeout(openRegister, 150);
}

function switchToLogin() {
    closeRegister();
```

```
    setTimeout(openLogin, 150);
}

async function doLogin() {
    const user = document.getElementById('loginUser').value;
    const pass = document.getElementById('loginPass').value;
    const err = document.getElementById('loginError');
    if (!user || !pass) { err.textContent = 'Заполните все поля';
err.classList.add('show'); return; }
    const res = await fetch('/api/login', { method: 'POST', headers:
{'Content-Type':'application/json'}, body:
JSON.stringify({username:user, password:pass}) });
    if (res.ok) {
        const data = await res.json();
        token = data.token;
        localStorage.setItem('token', token);
        closeLogin();
        showLoggedInUI();
        loadCart();
        applyFilters();
    } else {
        err.textContent = 'Неверный логин или пароль';
        err.classList.add('show');
    }
}

async function doRegister() {
    const user = document.getElementById('regUser').value;
    const pass = document.getElementById('regPass').value;
    const err = document.getElementById('registerError');
    if (!user || !pass) { err.textContent = 'Заполните все поля';
err.classList.add('show'); return; }
    const res = await fetch('/api/register', { method: 'POST',
headers: {'Content-Type':'application/json'}, body:
JSON.stringify({username:user, password:pass}) });
    if (res.ok) {
        alert('Регистрация успешна! Теперь войдите.');
        closeRegister();
        document.getElementById('loginUser').value = user;
        document.getElementById('loginPass').value = pass;
        setTimeout(openLogin, 200);
    } else {
        err.textContent = 'Пользователь с таким логином уже
существует';
        err.classList.add('show');
    }
}
```

Файл «static/style.css»

```
* {
    margin: 0;
```

```
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI',
Roboto, sans-serif;
    background: #f5f5f5;
}

/* ===== HEADER ===== */
header {
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: white;
    padding: 1rem 2rem;
    display: flex;
    justify-content: space-between;
    align-items: center;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}

header h1 {
    font-size: 24px;
    cursor: pointer;
}

/* ===== КНОПКИ АВТОРИЗАЦИИ ===== */
.login-btn, .register-btn {
    padding: 10px 20px;
    border: none;
    border-radius: 10px;
    font-weight: 600;
    font-size: 14px;
    cursor: pointer;
    transition: 0.3s;
}

.login-btn {
    background: white;
    color: #667eea;
}
.login-btn:hover {
    background: #f0f0f0;
    transform: translateY(-2px);
}

.register-btn {
    background: rgba(255,255,255,0.2);
    color: white;
    border: 2px solid white;
}
.register-btn:hover {
```

```
    background: white;
    color: #667eea;
    transform: translateY(-2px);
}

.logout-btn {
    padding: 8px 16px;
    background: rgba(255,255,255,0.2);
    color: white;
    border: 1px solid white;
    border-radius: 8px;
    cursor: pointer;
    font-weight: 600;
    transition: 0.3s;
}
.logout-btn:hover {
    background: white;
    color: #667eea;
}

#user-info {
    display: none;
    align-items: center;
    gap: 10px;
    color: white;
    font-weight: 500;
}

/* ===== МОДАЛЬНЫЕ ОКНА ===== */
#login-modal, #register-modal {
    display: none !important;
    position: fixed !important;
    top: 0 !important;
    left: 0 !important;
    width: 100% !important;
    height: 100% !important;
    background: rgba(0, 0, 0, 0.5) !important;
    backdrop-filter: blur(8px) !important;
    justify-content: center !important;
    align-items: center !important;
    z-index: 9999 !important;
    animation: fadeIn 0.3s ease !important;
}

#login-modal.active, #register-modal.active {
    display: flex !important;
}

@keyframes fadeIn {
    from { opacity: 0; transform: scale(0.95); }
    to { opacity: 1; transform: scale(1); }
}
```

```
.modal-box {
    background: white !important;
    padding: 40px 35px !important;
    border-radius: 20px !important;
    max-width: 400px !important;
    width: 90% !important;
    box-shadow: 0 20px 60px rgba(0,0,0,0.3) !important;
    position: relative !important;
    margin: auto !important;
}

.modal-box .close-btn {
    position: absolute;
    top: 15px;
    right: 20px;
    font-size: 24px;
    background: none;
    border: none;
    cursor: pointer;
    color: #999;
    transition: 0.3s;
}
.modal-box .close-btn:hover {
    color: #333;
    transform: rotate(90deg);
}

.modal-box h2 {
    text-align: center;
    margin-bottom: 8px;
    font-size: 26px;
    color: #333;
}

.modal-box .subtitle {
    text-align: center;
    color: #888;
    font-size: 14px;
    margin-bottom: 25px;
}

.modal-box label {
    display: block;
    font-size: 14px;
    font-weight: 600;
    color: #444;
    margin-top: 15px;
    margin-bottom: 5px;
}

.modal-box input {
```

```
    width: 100%;
    padding: 12px 15px;
    border: 2px solid #e0e0e0;
    border-radius: 10px;
    font-size: 15px;
    transition: 0.3s;
    background: #f9f9f9;
    box-sizing: border-box;
}

.modal-box input:focus {
    border-color: #667eea;
    outline: none;
    background: white;
    box-shadow: 0 0 0 4px rgba(102, 126, 234, 0.15);
}

.modal-box .btn-primary {
    width: 100%;
    padding: 14px;
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    color: white;
    border: none;
    border-radius: 10px;
    font-size: 16px;
    font-weight: 600;
    cursor: pointer;
    margin-top: 20px;
    transition: 0.3s;
}

.modal-box .btn-primary:hover {
    transform: translateY(-2px);
    box-shadow: 0 8px 25px rgba(102, 126, 234, 0.4);
}

.modal-box .btn-success {
    background: linear-gradient(135deg, #28a745 0%, #20c997 100%);
}

.modal-box .btn-success:hover {
    box-shadow: 0 8px 25px rgba(40, 167, 69, 0.4);
}

.modal-box .switch-link {
    text-align: center;
    margin-top: 15px;
    font-size: 14px;
    color: #888;
}

.modal-box .switch-link a {
```

```
    color: #667eea;
    font-weight: 600;
    cursor: pointer;
    text-decoration: none;
}

.modal-box .switch-link a:hover {
    text-decoration: underline;
}

.modal-box .error-msg {
    background: #fee2e2;
    color: #dc2626;
    padding: 10px 14px;
    border-radius: 8px;
    font-size: 14px;
    margin-top: 10px;
    display: none;
    text-align: center;
}

.modal-box .error-msg.show {
    display: block;
}

/* ===== ОСНОВНАЯ СЕТКА ===== */
main {
    display: flex;
    gap: 20px;
    padding: 20px;
    max-width: 1400px;
    margin: 0 auto;
}

/* ===== САЙДБАР ===== */
.sidebar {
    width: 260px;
    background: white;
    padding: 15px;
    border-radius: 10px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    height: fit-content;
    position: sticky;
    top: 20px;
}

.sidebar h3 {
    margin: 15px 0 10px 0;
    color: #333;
    font-size: 16px;
}
.sidebar h3:first-of-type {
```

```
    margin-top: 0;
}

.sidebar button {
    display: block;
    width: 100%;
    padding: 8px 12px;
    margin: 5px 0;
    background: #f0f0f0;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    text-align: left;
}
.sidebar button:hover {
    background: #667eea;
    color: white;
}

#search-input {
    width: 100%;
    padding: 8px;
    border-radius: 5px;
    border: 1px solid #ddd;
    margin-bottom: 10px;
}

.price-inputs {
    display: flex;
    gap: 10px;
    margin-bottom: 10px;
}
.price-inputs input {
    width: 50%;
    padding: 6px;
    border-radius: 5px;
    border: 1px solid #ddd;
}

#brands-list {
    max-height: 200px;
    overflow-y: auto;
    margin-bottom: 10px;
}
#brands-list label {
    display: block;
    margin: 5px 0;
    cursor: pointer;
    font-size: 14px;
}
#brands-list input {
    margin-right: 8px;
```

```
}

/* ===== КОНТЕНТ ===== */
.content {
    flex: 1;
}

.products-grid {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(220px, 1fr));
    gap: 20px;
    margin-top: 15px;
}

.product-card {
    background: white;
    border-radius: 10px;
    padding: 15px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    transition: transform 0.2s, box-shadow 0.2s;
    cursor: pointer;
}
.product-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 5px 20px rgba(0,0,0,0.15);
}
.product-card h3 {
    font-size: 16px;
    margin-bottom: 8px;
    color: #333;
}
.product-card .brand {
    color: #888;
    font-size: 12px;
    margin-bottom: 10px;
}
.product-card .price {
    font-size: 20px;
    font-weight: bold;
    color: #667eea;
    margin: 10px 0;
}
.product-card button {
    width: 100%;
    padding: 8px;
    background: #667eea;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    margin-top: 5px;
}
```

```
.product-card button:last-of-type {
    background: #28a745;
}

/* ===== КОРЗИНА ===== */
.cart-sidebar {
    width: 300px;
    background: white;
    padding: 15px;
    border-radius: 10px;
    box-shadow: 0 2px 5px rgba(0,0,0,0.1);
    position: sticky;
    top: 20px;
    height: fit-content;
}
.cart-sidebar h3 {
    margin-bottom: 15px;
    color: #333;
}
.cart-item {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px 0;
    border-bottom: 1px solid #eee;
}
.cart-item button {
    background: #ff4757;
    color: white;
    border: none;
    border-radius: 3px;
    padding: 2px 8px;
    cursor: pointer;
}
#cart-total {
    margin-top: 15px;
    font-weight: bold;
    font-size: 18px;
    text-align: right;
}

/* ===== БЕЙДЖИ ===== */
.filter-badge {
    display: inline-block;
    background: #667eea20;
    padding: 4px 12px;
    border-radius: 20px;
    font-size: 13px;
    margin: 3px;
}
.clear-all-btn {
    background: #dc3545;
```

```
    color: white;
    border: none;
    padding: 4px 12px;
    border-radius: 20px;
    cursor: pointer;
    margin-left: 10px;
}

.recommendations-banner {
    background: linear-gradient(135deg, #667eea20, #764ba220);
    padding: 20px;
    border-radius: 10px;
    margin-bottom: 30px;
}

.empty-state {
    text-align: center;
    padding: 50px;
    color: #888;
}

/* ===== АДАПТИВ ===== */
@media (max-width: 1024px) {
    main {
        flex-direction: column;
    }
    .sidebar {
        width: 100%;
        position: static;
    }
    .cart-sidebar {
        width: 100%;
        position: static;
    }
    .products-grid {
        grid-template-columns: repeat(auto-fill, minmax(180px,
1fr));
    }
    header {
        flex-wrap: wrap;
        gap: 10px;
    }
    #auth-section {
        display: flex;
        flex-wrap: wrap;
        justify-content: center;
    }
}

@media (max-width: 480px) {
    header h1 {
        font-size: 18px;
```

```
    }
    .login-btn, .register-btn, .logout-btn {
        padding: 6px 12px;
        font-size: 12px;
    }
    .products-grid {
        grid-template-columns: repeat(auto-fill, minmax(150px,
1fr));
    }
    .modal-box {
        padding: 25px 20px;
    }
```

Приложение В

(обязательное)

Интернет-магазин электронной техники с интеллектуальным анализом предпочтений клиентов

Презентация

Интернет-магазин электроники с интеллектуальным анализом предпочтений

Автор работы:
студент гр. 22ВВП2 Тельнов Илья Владимирович

1

Цель работы

Цель работы: Цель работы заключается в разработке интернет-магазина электроники, который предоставляет пользователю возможность просматривать каталог товаров с фильтрацией по категориям, цене и брендам, управлять корзиной, а также получать персонализированные рекомендации на основе интеллектуального анализа поведенческих данных и неявных предпочтений.

2

Аналоги:
Раздел электроники Wildberries
Раздел электроники Ozon
М.Видео
DNS
Citilink

4

Актуальность и аналоги

Актуальность: актуальность темы обусловлена противоречием между:

- возросшими ожиданиями пользователей к персонализации и скоростью выбора в интернет–магазинах электроники;
- недостаточной развитостью моделей интернет–магазинов, способных в реальном времени анализировать неявные предпочтения и адаптировать выдачу товаров.

3

Инструменты разработки

- **Бэкенд:** язык Python 3.11+, веб-фреймворк FastAPI, ASGI-сервер Uvicorn, SQLAlchemy ORM, Pydantic для валидации данных.
- **Машинное обучение:** Pandas, NumPy для обработки данных; Scikit-learn для вычисления косинусной близости; алгоритм ALS (Alternating Least Squares) для коллаборативной фильтрации.
- **Фронтенд:** HTML5, CSS3, JavaScript с Fetch API для асинхронных запросов.
- **Контейнеризация:** Docker и Docker Compose.

5

Архитектура

- **Архитектура** данной системы клиент-серверная двухуровневая.
- **Нижний уровень (клиент)** — веб-интерфейс на HTML/CSS/JS, который обеспечивает:
- Просмотр каталога с фильтрацией
- Карточку товара с таймером просмотра
- Корзину
- Персонализированные рекомендации
- **Верхний уровень (сервер)** — REST API на FastAPI, модуль интеллектуального анализа и работа с базой данных через SQLAlchemy.
- Передача данных осуществляется по HTTP-протоколу в формате JSON.

6

Таблицы:
users — пользователи (id, username, password)
products — товары (id, name, category, price, brand, description, image_url)
events — поведенческие события (id, user_id, product_id, event_type, duration_seconds, timestamp)
cart — корзина (id, user_id, product_id, quantity)

7

Принцип работы:
Сбор неявных поведенческих данных
Построение матрицы user-item
Вычисление косинусной близости между товарами
Формирование персонализированных рекомендаций
Рекомендации формируются на основе топ-3 товаров пользователя, при этом уже просмотренные товары исключаются. Для новых пользователей реализовано решение холодного старта — показываются популярные товары.

Результаты работы

9

Спасибо за внимание!

10